University of Alberta

Exploring several techniques for QRS Detection, Feature Extraction and Classification.

By

Marlene Carolina Rodriguez de Ramirez ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of
the requirements for the degree of Master of Science

Department of Electrical and Computer Engineering

Edmonton, Alberta
Spring, 2004

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-96543-0
Our file Notre référence
ISBN: 0-612-96543-0

The author has granted a non-exclusive license allowing the Library and Archives Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Several techniques of QRS detection, feature selection and classification are studied in this work.

QRS detection was performed with: 1)Okada's digital filter, 2)the MOBD algorithm by Suppappola and Sun, 3) an enhanced version of Okada's method, and 4)an artificial neural network algorithm. Feature selection involved principal component analysis and Hermite transform. For comparison, experiments were also carried out using the samples of QRS segments. Classification was completed by using: 1)linear discriminant analysis (LDA); 2)self-organizing maps (SOM), 3)radial basis function (RBF) neural networks using orthogonal least squares (OLS), and 4)hybrid SOM-RBF neural networks.

The experiments were carried out on extracts of the MIT-BIH database. Four classes were used for the classification study: Normal, Premature-ventricular-contraction, Right-bundle-branch-block, and Left-bundle-branch-block.

For QRS detection, the neural network had the best accuracy.

The best classification results, in terms of accuracy and kappa score in testing, were obtained with the hybrid SOM-RBF using the samples of QRS segments.

I can do everything through Christ who gives me strength

Phil 4:13

Dedicated to the treasures of my life:

My beloved husband, Lino
And
My cherished child, Cristian Alejandro.

Acknowledgement

All praise be given to the Lord my God, Father of us all, for I am only an instrument of his Greatness.

I would like to thank my supervisor, Dr. Pedrycz for his advice, his patience and constant support over the years of my graduate studies.

My infinite love goes to my beloved husband, Lino; here we are living our dream. Thank you for being always by my side encouraging me with your loving support at all times.

To my baby, Cristian, you are the light of our lives, the smile that warms my heart whenever I feel down. You are the happiness of our home. Thanks to God for you, our treasure.

To my parents, Marlene and Abbes: Words cannot tell how grateful I am to be your child. Thank you for your boundless love, your guidance, and your prayers.

To my parents in law, especially to my second mother Zoraida. Thank you for your love and sacrifice. Without your help, this job would not have been finished any time soon.

To my dear friends: the family that we choose. Thanks to Milene, Nelly, mamá Cristina, Nasim, Brigida, Ozgur, Ivonne, Ruben, and Alex, for your friendship, encouragement and unconditional help.

Last but far from least, to the brothers Knights of Columbus and their families, thank you for your caring and loving support. My special gratitude goes to Eugene and Daria, who patiently helped me to improve the grammar and vocabulary of this document; and to Ken, Marianne, Cleira, Darrel, Marielos and Silvio, for taking our child into their loving care when we most needed it.

Table of contents

1. INTRODUCTION 1

2. PROBLEM OVERVIEW 3

2.1. INTRODUCTION 3

2.2. PROBLEM STATEMENT AND OBJECTIVES 3

2.3. MEDICAL BACKGROUND 4

2.3.1. The heart, a brief description 4

2.3.2. Notions of Electrocardiography 9

2.4. MATHEMATICAL BACKGROUND 22

2.4.1. Linear regression 23

2.4.2. Normalization 24

2.4.3. Entropy 29

2.4.4. Measures of performance 31

2.4.5. Measures of fit 35

2.5. ECG AUTOMATIC ANALYSIS 37

2.5.1. Elimination or reduction of the baseline wandering 37

2.5.2. Modeling the QRS 39

2.5.3. QRS Detection 49

2.5.4. Classification of the QRS complex; related work 80

2.6. DESCRIPTION OF THE DATA SET 86

2.6.1. The MIT-BIH arrhythmia Database 87

2.6.2. Data set for the study on QRS detection 90

2.6.3. Data set for the study on QRS classification 95

2.6.4. Data set for features evaluation 97

3. CLASSIFICATION METHODS 98

3.1. INTRODUCTION 98

3.2. LINEAR DISCRIMINANT ANALYSIS (LDA) 98

3.3. SELF ORGANIZING MAPS (SOM) 108

3.4. RADIAL BASIS FUNCTIONS NEURAL NETWORKS (RBF NN) 114

3.5. PROPOSED HYBRID RBF NEURAL NETWORKS FOR CLASSIFICATION PURPOSES .. 121

4. EXPERIMENTAL STUDY 127

4.1. INTRODUCTION 127

4.2. FEATURE EXTRACTION AND SELECTION 130

4.2.1. Feature extraction and selection using PCA 130

4.2.2. Feature extraction and selection using Hermite Transformation 134

4.3. QRS DETECTION 138

4.3.1. Okada QRS detector 138

4.3.2. MOBD QRS detector 143

4.3.3. Neural networks QRS detector 147

4.3.4. Summary of QRS detection results and discussion 151

4.4. QRS CLASSIFICATION 154

4.4.1. SOM as classifier 154

4.4.2. RBF-OLS 167

4.4.3. LDA 175

4.4.4. Proposed classifier SOM-RBF 176

4.4.5. Summary of the best classification results 185

5. CONCLUSION AND RECOMMENDATIONS FOR FUTURE WORK 186

6. BIBLIOGRAPHY 188

List of tables

Table 2-1: Standard electrocardiogram leads [60] *10*

Table 2-2: Values that each feature can take on the example (Data set consisting of ten 4-dimensional vectors). *24*

Table 2-3: Scaled version of the example data set *26*

Table 2-4: z-score version of the example data set *27*

Table 2-5: logistic version of the example data set *29*

Table 2-6: Confidence of Kappa Scores [61] *34*

Table 2-7: Example of Confusion matrix *34*

Table 2-8: Location of the R peaks by the annotator and by the Okada detector *54*

Table 2-9: Position of the detected and the labeled peaks using Okada's with a signal having a stepwise baseline drift *55*

Table 2-10: Location of the R peaks by the annotator and by the Pan-Tompkins/Hamilton-Tompkins detector *59*

Table 2-11: Position of labeled and detected peaks in the case of a baseline drift using the Hamilton-Tompkins/Pan-Tompkins's method. *61*

Table 2-12: Location of the R peaks by the annotator and by the MOBD detector in the first example. *66*

Table 2-13: Position of labeled and detected peaks in the case of a baseline drift using the MOBD method *67*

Table 2-14: Location of the R peaks by the annotator and by the neural networks detector in the first example. *78*

Table 2-15: Location of the R peaks by the annotator and by the neural networks detector in the case of a stepwise baseline drift. *79*

Table 2-16: Record numbers of the 100-series of the MIT-BIH arrhythmia database *87*

Table 2-17: Record numbers of the 200-series of the MIT-BIH arrhythmia database *87*

Table 2-18: Beat types of the MIT-BIH arrhythmia database *88*

Table 2-19: Distribution of classes for the extract used to study QRS detection algorithms *90*

Table 2-20: Distribution of classes per record; 100-series *91*

Table 2-21: Distribution of classes per record; 200-series *92*

Table 3-1: Several forms of basis functions considered in the lecture *115*

Table 3-2: Type of distance measures *116*

Table 4-1: Classification accuracy and kappa score using LDA depending on the number of Hermite coefficients *137*

Table 4-2: Results in testing the original Okada QRS detector *139*

Table 4-3: Results using the improved version of the Okada QRS detector *140*

Table 4-4: Positive predictivity of the Okada QRS detector per class *142*

Table 4-5: Results using the MOBD QRS detector *144*

Table 4-6: Distribution of classes of records sorted by error rate with the MOBD algorithm *146*

Table 4-7: Results using the Neural networks QRS detector trained with option1 .... *148*

Table 4-8: Results using the Neural networks QRS detector trained with option2 .... *149*

Table 4-9: Summary of results for QRS detection *152*

Table 4-10: Average entropy per class in the SOM classifier for the Hermite transformed data set with no normalization. *155*

Table 4-11: Average entropy per class in the SOM classifier for the Hermite transformed data set with logistic normalization *156*

Table 4-12: Average entropy per class in the SOM classifier for the data set formed by QRS segments with no normalization. *157*
Table 4-13: Average entropy per class in the SOM classifier for the data set formed by QRS segments with logistic normalization. *158*
Table 4-14: Average entropy in the SOM classifier depending on the type of data set. *159*
Table 4-15: Accuracy of the SOM classifier in training *160*
Table 4-16: Accuracy of the SOM classifier in testing *161*
Table 4-17: Kappa score of the SOM classifier in training *162*
Table 4-18: Kappa score of the SOM classifier in testing *162*
Table 4-19: Confusion matrix for training and testing using the SOM classifier 30x30 (QRS segments with logistic normalization) *167*
Table 4-20: Results using the RBF-OLS classifier with 4, 250 and 900 centers on the QRS segments with no normalization *170*
Table 4-21: Results using the RBF-OLS classifier with 4, 250 and 900 centers on the QRS segments with logistic normalization *172*
Table 4-22: Results using the RBF-OLS classifier with 4, 250 and 900 centers on the Hermite coefficients data with no normalization *173*
Table 4-23: Results using the RBF-OLS classifier with 4, 250 and 900 centers on the Hermite Coefficients data with logistic normalization *174*
Table 4-24: Accuracy with the LDA classifier *175*
Table 4-25: Parameters adjusted on the proposed SOM-RBF classifier *178*
Table 4-26: Best result for the SOM-RBF classifier *179*
Table 4-27: Best results of SOM-RBF for the QRS segments with no normalization. *181*
Table 4-28: Best results of SOM-RBF for the QRS segments with logistic normalization *181*
Table 4-29: Best results of SOM-RBF for the Hermite coefficients data set with no normalization *182*
Table 4-30: Best results of SOM-RBF for the Hermite coefficients data set with logistic normalization *182*
Table 4-31: Value of the parameters that produced the best classification results. *183*
Table 4-32: Summary of the best classification results *185*

List of Figures

Figure 2-1: Flow of blood in the heart [60] 5
Figure 2-2: Electrical conduction system [33] 7
Figure 2-3: Timing on the electrical conduction process [33] 8
Figure 2-4: Depolarization and Repolarization process [33] 9
Figure 2-5: Location of the electrodes for the 12-lead ECG 10
Figure 2-6: Electrical basis of the ECG [33] 12
Figure 2-7: Possible directions of the normal QRS [33] 14
Figure 2-8: Depolarization sequence when there is a right bundle branch block [33] 16
Figure 2-9: Example of right bundle branch block beats (RBBB), compared to normal beats (N) 17
Figure 2-10: Depolarization sequence when there is a left bundle branch block [33] 18
Figure 2-11: Example of left bundle branch block beats (LBBB), compared to normal beats (N). 19
Figure 2-12: Example of the effects of muscle tremor on the recordings [33] 21
Figure 2-13: Example of the effect of patient movement [60] 21
Figure 2-14: Effect of a 60 cycle ground interference on the ECG recordings [60] 22
Figure 2-15: Example of the effect of loose electrodes on the ECG recordings [33] 22
Figure 2-16: Example of non-normalized values in each feature for a data set containing ten 4-dimensional vectors. 25
Figure 2-17: Example of normalized values in each variable for a data set containing 10 vectors, each with 4 features, using the range method of normalization. 26
Figure 2-18: Example of normalized values for each one of the 4 features of the data set containing 10 samples, using the z-score normalization. 28
Figure 2-19: Example of normalized values for each one of the 4 variables of the data set containing 10 samples, using the logistic normalization. 29
Figure 2-20: Example using four classes: Distribution of the classes vs. entropy. 31
Figure 2-21: Plots of the first eight Hermite basis functions 42
Figure 2-22: Block Diagram of AHMES [66]. 45
Figure 2-23: Flowchart of the algorithm proposed by Okada 51
Figure 2-24: Okada's method: An illustrative example. 53
Figure 2-25: Example of Okada's method response in the case of a baseline drift. 55
Figure 2-26: Flowchart of the Pan-Tompkins/Hamilton-Tompkins algorithm for QRS detection 57
Figure 2-27: Expected relationship of a QRS complex to the moving integration waveform, exploited to detect the fiducial points 58
Figure 2-28: Pan-Tompkins/Hamilton-Tompkins algorithm: an example 60
Figure 2-29: Example of Hamilton-Tompkins/Pan-Tompkins's method response in the case of a baseline drift 61
Figure 2-30: Flowchart of MOBD algorithm 63
Figure 2-31: Example of a 4th order MOBD implementation 65
Figure 2-32: Example of MOBD QRS detector when there is a stepwise baseline drift 66
Figure 2-33: shape of the logistic sigmoid for different values of λ 68
Figure 2-34: Configuration of the two-layer perceptron 70
Figure 2-35: Neural Network training process 74
Figure 2-36: Inputs/Output of the neural network. Example for a centered QRS 75
Figure 2-37: Desired Outputs for one example of training set 76
Figure 2-38: Result of the neural network on the training set of the example 77

Figure 2-39: Example Neural Networks Model tested on new data *78*
Figure 2-40: Example of neural network QRS detector when there is a stepwise baseline drift *79*
Figure 2-41: Distribution of classes per record; 100-series *93*
Figure 2-42: Distribution of classes per record; 200-series. *94*
Figure 2-43: R-R measurements for obtaining R^- *and* R^+ *96*
Figure 3-1: MDA-LDA example1: Projection of two-dimensional vectors onto two different lines [18] *99*
Figure 3-2: MDA-LDA example2: Definition in the original space of the projection lines y_1 *and* y_2 *in the direction of w1 and w2* *102*
Figure 3-3: MDA-LDA example2: Data set of two-dimensional vectors belonging to three classes in the original space (each sample is defined by coordinates in x_1 *and* x_2*)* *103*
Figure 3-4: MDA-LDA example2: Data in the transformed space (each sample is defined now by coordinates in y_1 *and* y_2*)* *103*
Figure 3-5: MDA-LDA example2: Discriminant functions for each class *107*
Figure 3-6: MDA-LDA example2: Classification results *107*
Figure 3-7: Two-dimensional self-organizing system *109*
Figure 3-8: Radii assignation (δ) *for two different neighborhood topologies: square and hexagonal* *110*
Figure 3-9: Configuration of the weights *112*
Figure 3-10: Location of the vectors equidistant to a reference vector using Euclidean, Hamming and Tschebyschev distances; example with two variables *116*
Figure 3-11: Structure of the Radial Basis Functions Network *117*
Figure 3-12: Subset of a data set formed by 2-dimensional vectors. *119*
Figure 3-13: Inputs/Outputs of the clustering process using SOM *123*
Figure 3-14: Inputs/Outputs of step 2 .... *124*
Figure 3-15: Clustering of the SOM based on entropy *126*
Figure 4-1: Experimental work *129*
Figure 4-2: % Fraction of Variance contributed by each principal component in the PCA *131*
Figure 4-3: Cumulative Variance when using different number of principal components for the PCA *131*
Figure 4-4: Squared Error when using a different number of principal components to approximate the original data set *132*
Figure 4-5: Correlation index vs. Number of principal components, averaged for all classes *133*
Figure 4-6: Accuracy with the LDA classifier when using different number of principal components for the PCA *134*
Figure 4-7: Number of Hermite coefficients vs. Squared error between the modeled and the original signals *135*
Figure 4-8: Number of Hermite coefficients vs. linear correlation coefficient between the modeled and the original signals *136*
Figure 4-9: Number of Hermite coefficients vs. Accuracy and Kappa score using LDA *137*
Figure 4-10: QRS detection error rate for each record using the original Okada algorithm and its improved version *143*
Figure 4-11: Positive predictivity and error rate for the MOBD algorithm *145*
Figure 4-12: Average error rate QRS detection for all the implemented algorithms *152*
Figure 4-13: Average accuracy QRS detection for all the implemented algorithms *152*
Figure 4-14: Best error rate per record (100-series) for each QRS detection algorithm ... *153*
Figure 4-15: Best error rate per record (200-series) for each QRS detection algorithm ... *153*

Figure 4-16: Average entropy per class vs. size of SOM map. Data set of Hermite coefficients, with no normalization 156
Figure 4-17: Average entropy per class vs. size of SOM map. Data set of Hermite coefficients, with logistic normalization 157
Figure 4-18: Average entropy per class vs. size of SOM map. Data set of QRS segments, with no normalization 158
Figure 4-19: Average entropy per class vs. size of SOM map. Data set of QRS segments, with logistic normalization 159
Figure 4-20: Size of the SOM vs. accuracy of the SOM classifier in training 160
Figure 4-21: Size of the SOM vs. accuracy in testing 161
Figure 4-22: Size of the SOM vs. kappa score in training 162
Figure 4-23: Size of the SOM vs. kappa score in testing 163
Figure 4-24: SOM classifier 30x30. Cluster map showing the winning class per neuron ..164
Figure 4-25: SOM classifier 30x30. Cluster map with entropy per neuron 166
Figure 4-26: Accuracy in training of RBF-OLS classifier depending on the number of centers. 168
Figure 4-27: Accuracy in testing of RBF-OLS classifier depending on the number of centers 169
Figure 4-28: Best results in training of SOM-RBF classifier depending on the size of the SOM. 180
Figure 4-29: Best results in testing of SOM-RBF classifier depending on the size of the SOM. 180
Figure 4-30: Number of clusters vs. size of the SOM 181
Figure 4-31: clustering map of the SOM 20x20, with the winning class per neuron, and the cluster number assigned by the software. 184
Figure 4-32: Distribution map of the SOM 20x20, with the winning class per neuron, and the cluster number assigned by the software 185

1. Introduction

Electrocardiogram (ECG) analysis is a vast subject that is open to many research topics. The ECG is composed of several characteristic waves that represent different stages in the cardiac rhythm. Because the heart activity is cyclical, these waves appear repetitively in the ECG signal. Depending on the cardiac process of interest, the ECG's waves can be studied separately or altogether.

In this work, the set of waves of interest is the so-called QRS complex (formed by the Q, the R and the S waves). Two main problems can be identified when the patterns they form are studied: 1) detecting the QRS, and 2) classifying it once it has been detected.

In this work, several techniques are explored to achieve the goals of detecting and classifying of the QRS complex.

In the case of QRS detection, typical methods are described and some were implemented in this study: the pioneer digital filter introduced by Okada in 1979 [55] (which is a very comprehensive algorithm), the algorithm presented by Suppappola and Sun [77] that involves non-linear transformations of the ECG signal called the multiplication of backward differences or MOBD (which is known to give fast results), the algorithm presented by Tompkins et al. [29][57] (which is known to be very accurate), and an algorithm involving a neural networks configuration based on the work of Garcia et al. [21] (which allows to include learning capability to the QRS detector). Additionally, an enhanced version of Okada's method was developed. In this new method, the improved thresholding of the MOBD is incorporated into Okada's approach.

QRS classification was performed considering four main classes of beats: normal, premature ventricular contraction, left bundle branch block, and right bundle branch block (labeled as N, V, L, and R respectively). The classification problem was treated under the assumption that the time of occurrence of the QRS was known (i.e. the data set was formed by QRS segments rather than the complete ECG signal).

In this work four classifiers were tested:

- linear discriminant analysis (LDA),
- self-organizing maps (SOM) adapted to be used as classifiers,
- radial basis functions neural networks with orthogonal least squares (RBF-OLS NN), and
- a hybrid configuration obtained by the combination of the SOM clustering method with the RBF NN (SOM-RBF).

In this thesis, the first chapter (Chapter 2, Problem Overview) offers an overview of the problem, including a comprehensive medical background, mathematical background, background on ECG analysis, and the description of the data set that is used on the experiments (an extract of the MIT-BIH database [53]) to validate the different algorithms. Chapter 3, Classification Methods, is dedicated to the description of the classification methods that are implemented in this work. It also includes a Section about feature extraction of the QRS signal using principal component analysis (PCA) and Hermite transformation. The experimental study is presented in Chapter 4, Experimental Study. Finally, Chapter 5, Conclusion and Recommendations for future work, presents the conclusions from this work and some recommendations for future work.

2. Problem Overview

2.1. Introduction

In this Section, the problems to be solved as well as the objectives of the thesis are presented in Section 2.2, followed by Section 2.3, which includes a brief description of the operation of the heart and some notions of electrocardiography. The mathematical concepts used in this thesis are explained in Section 2.4. A review of previous work on ECG analysis is presented in Section 2.5, showing a range of options and related work present in the literature for eliminating or reducing of the ECG baseline wandering, modeling of the QRS, and classifying the QRS. Finally is presented the description of the data sets used for the different experiments in Section 2.6.

2.2. Problem Statement and objectives

Given the digitized version of an electrocardiographic signal, it is sought to explore among several neural networks techniques in order to design a classifier that may distinguish between several classes of beats. Some classification techniques have to be experimented with to construct a classifier with the highest classification rate.

There are many possible classes of beats to be studied. This work is limited to the four most common types of beats found in the MIT-BIH database [53]: normal beats (N), premature ventricular beats (V), right branch block beats (R) and left branch block beats (L).

The study involves, not only constructing the most suitable classifier in the group of study but also pre-processing the signal, detecting the beats to be classified, and exploring significant features to discriminate among classes. In other words, some basic procedures included in the process of pattern recognition are required to be performed to complete this task.

The ideal pre-processing of the signal would be the one that standardizes how the signal looks like, so that the differences between samples are only due to the difference in class, not the quality of the signal. The simplest way to do it would be eliminating the components of the signal that may be considered as noise.

The detection of the beats is itself a complete research topic. In the present work, the theme concerning the detection of the beats is reviewed in terms of a comparative study of some of the techniques previously presented by a number authors [54] [72] [55] [77] [1] [57] [29] [2] [67] [47] [21] [76].

The objective of studying the viability of the features is to find a way to reduce the number of variables in the data set as well as trying to maintain or accentuate the differences between classes in the new domain.

Mathematical and medical insights are explored so that the origins (and characteristics) of the signals studied in this work can be more fully developed and understood.

2.3. *Medical background*

2.3.1. The heart, a brief description

A. Introduction [7][25][33][60][79]

The heart is the organ that pumps blood to the vascular system owing to the synchronized contractile properties of the myocardium. The heart consists of four chambers: two thin-walled atria at the apex, and two thick-walled ventricles at the base. The right atrium and ventricle function as a unit, constituting the so-called right heart. Correspondingly, the left heart refers to the left atrium and ventricle.

The flow of blood in the heart is shown in Figure 2-1. Two systemic veins, the inferior and superior Vena Cava, return the deoxygenated blood from the body to the right atrium, from where the blood subsequently enters the right ventricle through the tricuspid valve. The contraction of the right ventricle provides the necessary force to open the Pulmonary valve and circulate the blood through the Pulmonary artery; this artery carries the blood to the lungs for reoxygenation. The oxygenated blood is returned by the Pulmonary vein to the left atrium, from where it is sent to the left ventricle through the Mitral valve. The contraction of the left ventricle—the systole—delivers the blood to the Aorta, which distributes the oxygenated blood throughout the body, thus completing the circle of a heart beat.

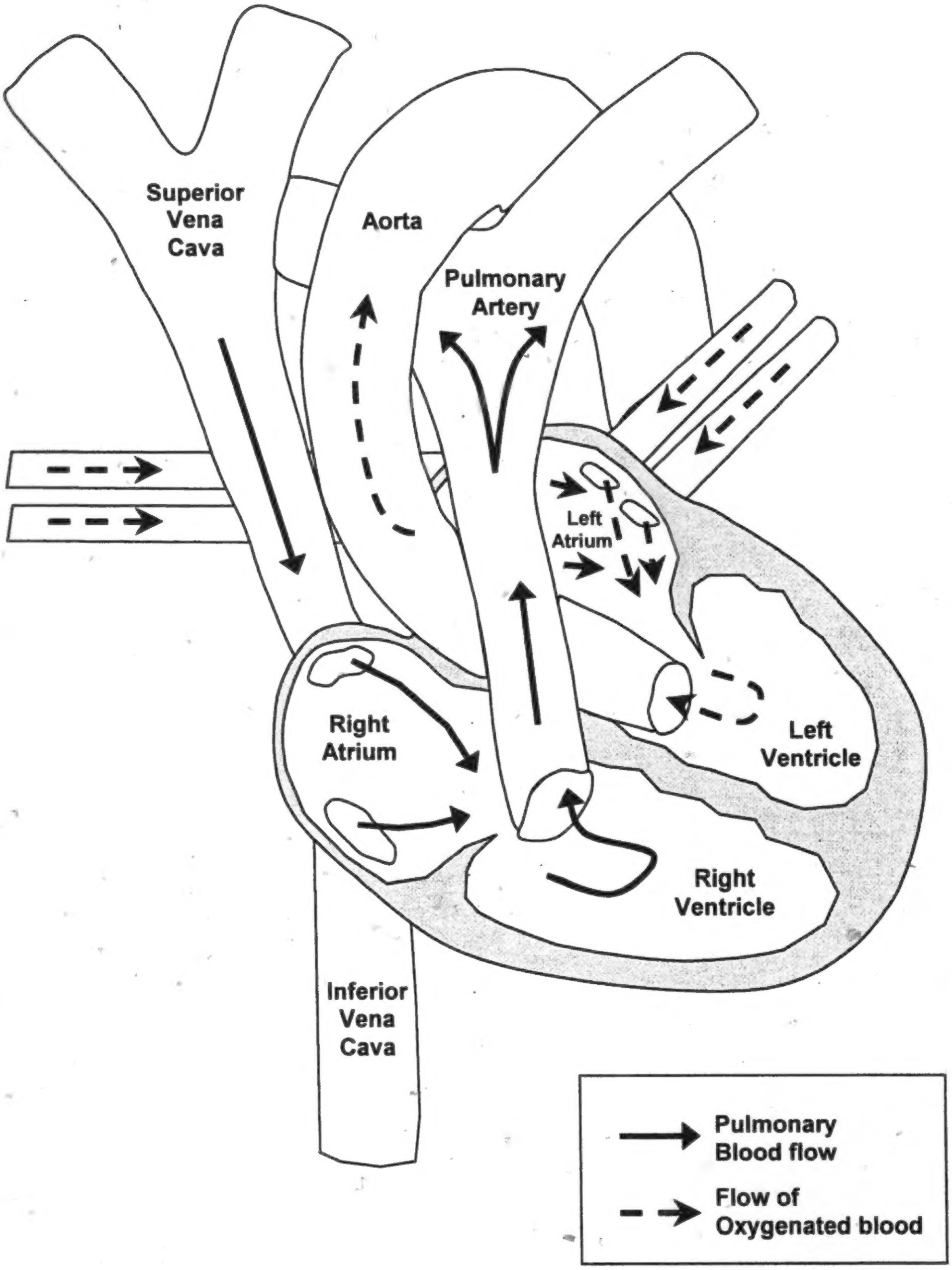

Superior Vena Cava
Aorta
Pulmonary Artery
Left Atrium
Right Atrium
Left Ventricle
Right Ventricle
Inferior Vena Cava
Pulmonary Blood flow
Flow of Oxygenated blood

Figure 2-1: Flow of blood in the heart [60]

B. Electrophysiology of the heart

Each heartbeat is the result of sequential activities, governed by electrochemical events. Cells that have the capacity to contract following an electrical stimulus constitute the cardiac muscles. These cells, called sarcomeres, are arranged in such a manner that once stimulated, they allow the contraction of the correspondent chamber. Figure 2-2 illustrates the electrical conduction system by which this process is executed.

Two separated clusters of sarcomeres electrically isolated from one another, form the atrial and the ventricular walls. Pacemaker cells lie in the wall of the right atrium forming the Sinoatrial (SA) node. This node automatically and regularly generates electrical impulses to stimulate the first cluster -the atrial sarcomeres. The internodal and the interatrial conduction tracts allow the impulses to spread through the atrium. When the impulses reach the junction between atria and ventricles, they stimulate the second cluster -the Atrioventricular (AV) node- that transmits the impulses in an orderly and timely manner to the ventricular sarcomeres. The impulses leave the AV node and enter the ventricles by the bundle of His (also called the common bundle) on the upper part of the interventricular septum. They continue then into the remaining part of the ventricles as the common bundle separates into a right branch and a left branch that run on either side of the septum. The impulses follow these branches, which subdivide consecutively to connect with the Purkinje network and then terminate at the muscle fibers of the myocardium.

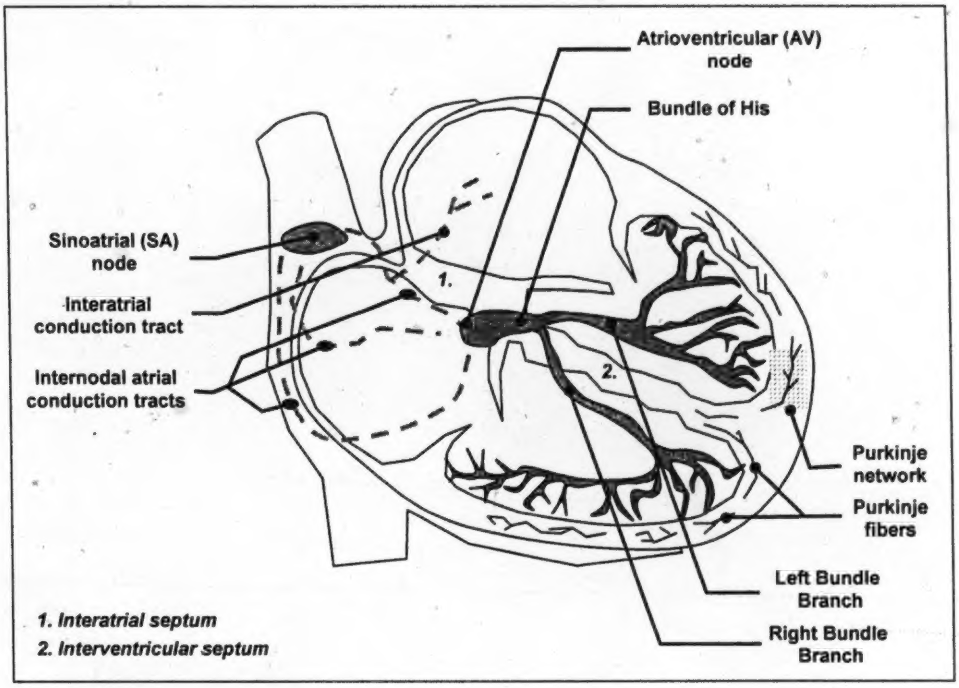
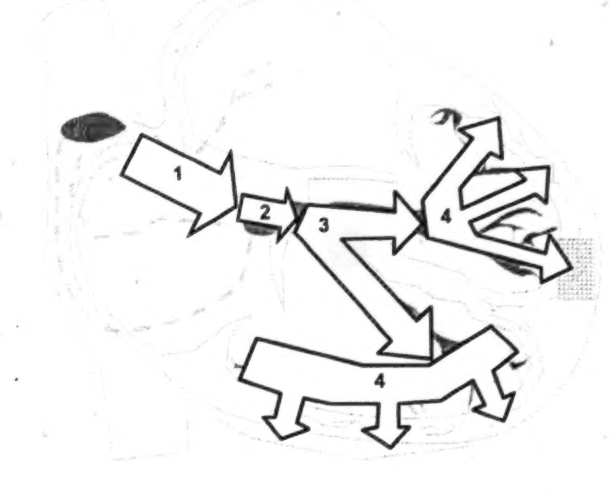

Atrioventricular (AV) node
Bundle of His
Sinoatrial (SA) node
Interatrial conduction tract
Internodal atrial conduction tracts
1.
2.
Purkinje network
Purkinje fibers
Left Bundle Branch
Right Bundle Branch
1. Interatrial septum
2. Interventricular septum

Figure 2-2: Electrical conduction system [33]

The timing for the impulses to travel through the electrical system of the heart is shown in Figure 2-3. The electrical impulses travel rapidly from the SA node to the AV node. From the AV node, however, they travel relatively slowly to reach the bundle of His, thus allowing the atria to contract and empty the blood into the ventricles before the occurrence of the ventricular contraction. The impulses travel more rapidly from the common bundle to the branches, which connect to the Purkinje network.



1. *SA node to AV node*: **0.03 sec**
2. *AV node to bundle of His*: **0.06 – 0.12 sec**
3. *Bundle of His to Bundle branches*: **0.03 – 0.05 sec**
4. *Bundle branches to Purkinje network*: **< 0.01 sec**

Figure 2-3: Timing on the electrical conduction process [33]

The electrochemical phenomenon that conducts the stimuli through the sarcomeres is called the Cardiac Action Potential. An action potential is the brief, rapid influx of positive ions across the cell membrane, through ion channels. The gradient in the concentration of such ions across the cell membrane is called the membrane potential.

A simple scheme of the process of depolarization and repolarization of a muscle fiber is shown in Figure 2-4. Note that the muscle fiber is an array of excitable cells.

Initially, muscle fibers (or sarcomeres) are negatively charged at rest (Figure 2-4a). When an electrical impulse arrives, the positive ions flow into the cell, thus depolarizing it. The depolarization of one cell generates an impulse of enough magnitude to depolarize the adjacent cells and then propagate the action potential, in this case, from left to right (Figure 2-4b). The muscle contracts where the polarity changes. The muscle fully contracts when all the sarcomeres are depolarized (Figure 2-4c). The repolarization then begins in the reverse direction (Figure 2-4d) until the muscle is fully repolarized and relaxed.

A cell can be stimulated to depolarize again even when it is not fully repolarized. This early depolarization can occur only if the cell has been repolarized enough to reach a certain membrane potential called the threshold potential.

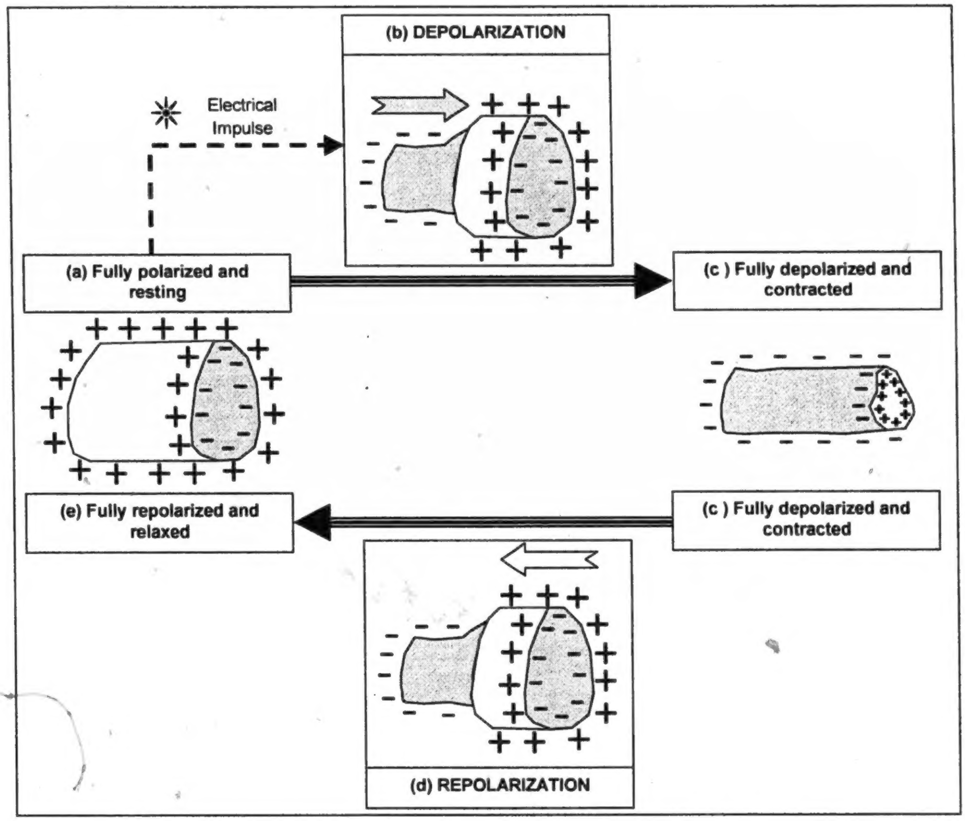

(b) DEPOLARIZATION
Electrical Impulse
(a) Fully polarized and resting
(c) Fully depolarized and contracted
(e) Fully repolarized and relaxed
(c) Fully depolarized and contracted
(d) REPOLARIZATION

Figure 2-4: Depolarization and Repolarization process [33]

2.3.2. Notions of Electrocardiography

The depolarization and repolarization of the atria and ventricles generate an electrical activity that can be recorded using surface electrodes attached to the skin at designated locations. The response that is generated is called an electrocardiogram (ECG); it has several deflections that represent some of the components of the cardiac cycle. The electrocardiogram shows the direction of the summated vector of the electrical activities occurring during each step of the cycle.

As mentioned above, these electrical vectors are recorded from surface electrodes. Each pair of electrodes of opposite polarity is called an ECG lead. Clinically, twelve leads are used to study the cardiac function: I, II, III, aVR, aVL, aVF, V_1, V_2, V_3, V_4, V_5, and V_6. Figure 2-5 and Table 2-1 [60] show the arbitrary location of each lead.

It is important to know the location of the lead in respect to the heart's position because the deflections on the electrocardiogram correspond with the direction of the depolarization or repolarization waves. For instance, if the depolarization wave moves toward the positive electrode of a lead, the deflection will be positive, and its amplitude will increment as the electrical current further faces the positive electrode.

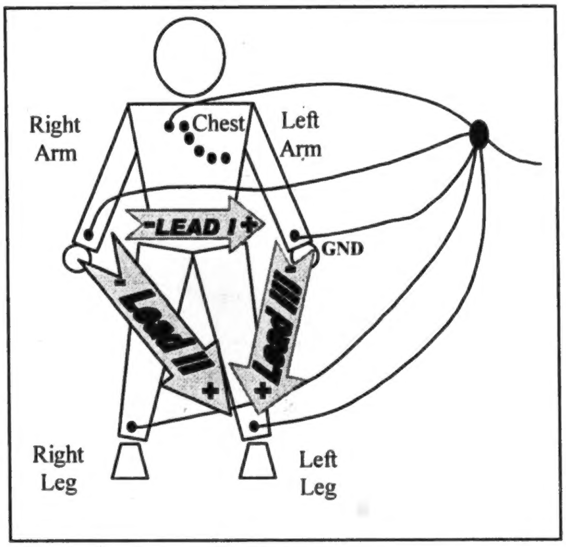


Figure 2-5: Location of the electrodes for the 12-lead ECG

Table 2-1: Standard electrocardiogram leads [60]

		Leads	Positive electrode	Negative electrode
Bipolar	1	I	Left arm	Right arm
	2	II	Left leg	Right arm
	3	III	Left leg	Left arm
Unipolar	4	aVR	Right arm	Central terminal*
	5	aVL	Left arm	
	6	aVF	Left leg	
Precordial	7	V_1	Right border of sternum in 4th Intercostal Space (ICS)	Central terminal*
	8	V_2	Left border of sternum in 4th ICS	
	9	V_3	Midway between V_2 and V_4	
	10	V_4	Midclavicular line 5th ICS	
	11	V_5	Midway between V_4 and V_6	
	12	V_6	Lateral chest in 5th ICS	
* The central terminal is a combination of electrode potentials, producing a summation effect.				

A. Basic components of an electrocardiogram

Lead II is commonly used to monitor the cardiac arrhythmias. The profile of the cardiac cycles components shown in Figure 2-6 correspond to the shape the electrocardiogram generally presents on lead II.

The depolarization of the atria is seen as the upward P wave (Figure 2-6a). The repolarization of the atria (Figure 2-6b) causes a negative deflection called the Ta wave, which is usually not visible in the electrocardiogram (Figure 2-6e) because the depolarization of the ventricles generates a stronger signal -the QRS complex- (Figure 2-6d) that masks it. Finally, the repolarization of the ventricles causes an upward wave -the T wave-, which appears after an isoelectric period (ideally, a period without any fluctuations of the electrical level).

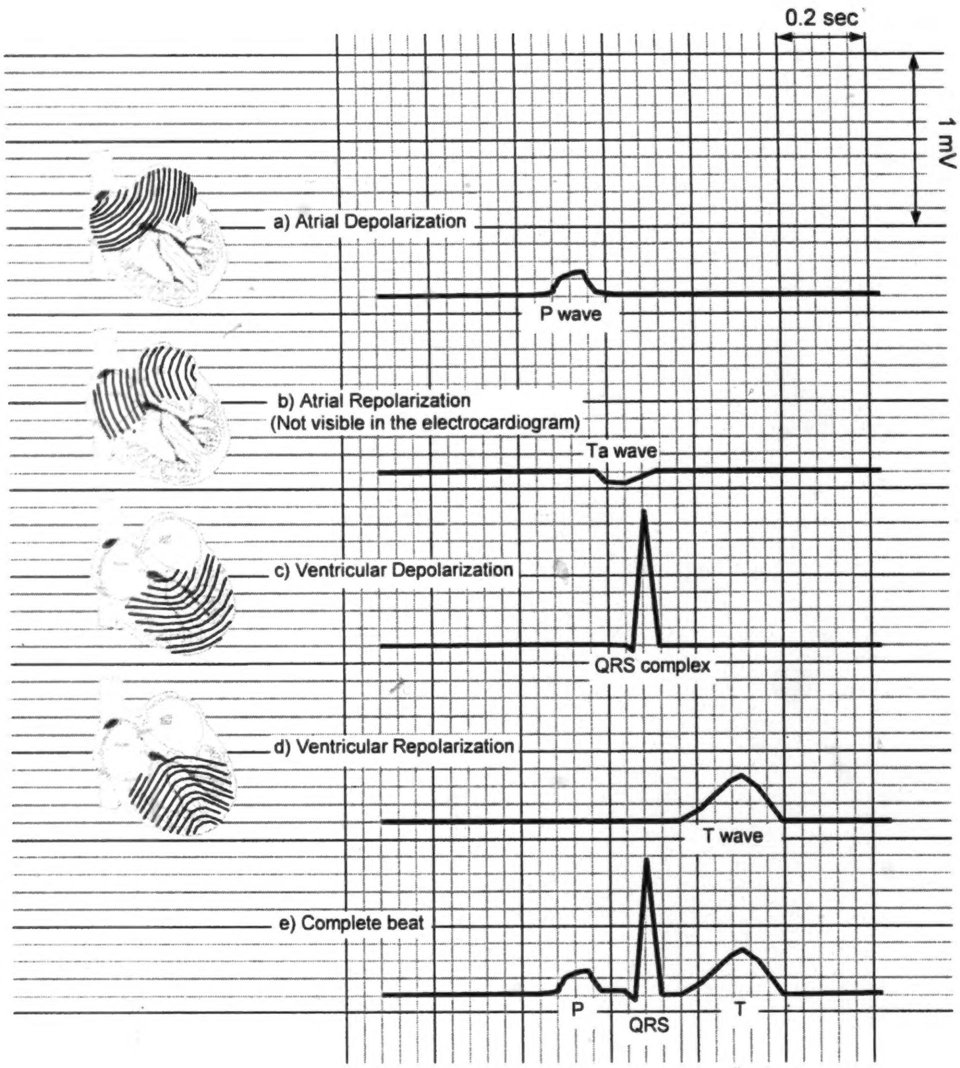

0.2 sec
1 mV
a) Atrial Depolarization
P wave
b) Atrial Repolarization
(Not visible in the electrocardiogram)
Ta wave
c) Ventricular Depolarization
QRS complex
d) Ventricular Repolarization
T wave
e) Complete beat
P
QRS
T

Figure 2-6: Electrical basis of the ECG [33]

The characteristics of the ECG components (P wave, QRS complex, and T wave) are as follows:

a. P wave

The P wave represents the depolarization of the atria. The upward deflection of the P wave is due to the right atrium depolarization vector, which faces Lead II positive electrode. As this depolarization vector moves away from the positive electrode, it forms the downward deflection of the P wave.

The normal P wave should have:

- an upright direction
- a duration of 0.10 seconds or less
- an amplitude between 0.05mV and 0.25 mV
- a smooth round shape

Additionally, the normal P wave should precede a QRS. The total duration of the normal P wave plus the duration from the end of the P wave to the beginning of the QRS complex (the length of the normal P-R interval) should be between 0.12 to 0.2 seconds.

b. QRS complex

The QRS complex is generated by the depolarization of the ventricles and consists of a group of consecutive waves: the Q wave, the R wave, and the S wave.

The **Q wave**, the initial negative deflection in the QRS complex, represents the depolarization of the interventricular septum.

The rest of the complex corresponds to the activation of the left and right ventricle, which happen simultaneously. The signal produced by the activation of the left ventricle, however, prevails over the one generated by the right ventricle because the left ventricle is of greater size.

Depending on the position of the heart in the thorax, the current generated during ventricular depolarization can flow toward the left leg generating the first positive deflection (the **R wave**) and then shift away from it generating a negative deflection called the **S wave**.

The normal QRS is not characterized by a unique direction. In effect, its direction can be predominantly positive, predominantly negative or biphasic as shown in Figure 2-7.

In the first case, the size of the R wave exceeds the size of the Q wave (identified as q because of its small size), and the S wave (which does not show in this case).

In the second case, the S wave is dominant upon the R wave (identified as r because of its small size) and upon the Q wave that is practically non-existent.

In the last case, the size of the Q wave could be comparable to the size of the R wave (And hence the identification of both waves with capital letters).

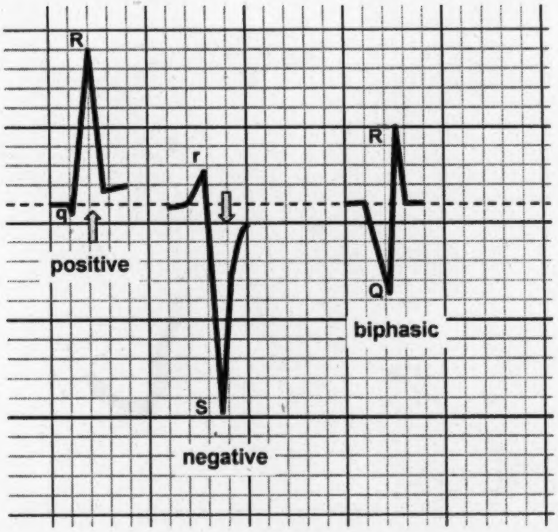


Figure 2-7: Possible directions of the normal QRS [33]

The amplitude of the components of the normal QRS complex is variable. In lead II, the amplitude of the R or S wave (taken from the baseline) may vary from a couple of millivolts to a dozen millivolts or more.

In adults, the duration of a Normal QRS is estimated to be from 60 to 100 milliseconds while QRS waves that last more than 120 ms are usually considered abnormal. Note that there is a gap between 100 ms and 120 ms in which the QRS could be normal or not depending on other factors.

c. T wave

As shown in Figure 2-6, there are two T waves that hold different directions: the Ta wave and the T wave.

The atrial T wave or Ta wave (shown in Figure 2-6b), whose direction is negative, is produced by the atrial repolarization. The Ta wave follows the P wave but it is usually not noticeable in the recordings of electrocardiogram, because the atria repolarization occurs at the same time than the ventricular depolarization (that produces the QRS, a much stronger signal than the Ta wave).

The ventricular T wave (called T wave, in short) is produced by the ventricular repolarization. When it is normal, it follows the QRS complex, its direction is usually positive, and its amplitude should not exceed 0.5 mV with respect to the baseline. The duration of the T wave usually varies between 100 ms and 250 ms, sometimes more. The characteristics of this wave when it is abnormal are variable:

- Its direction could be positive, negative of even biphasic
- Its amplitude may or may not exceed normal ranges
- Its duration coincides with the ranges of the normal T wave.

B. Abnormal beats; brief description

Many characteristics in the beats may lead to classify them as being abnormal. There is an extensive number of classes of abnormal beats [33], [60]. In this Section, only the three classes of interest to this work are described: the premature ventricular contraction, the right bundle branch block and the left bundle branch block.

a. Premature Ventricular Contraction

The premature ventricular contraction (PVC) also called ventricular premature contraction (VPC) is defined in [60] as "[…] a beat occurring early in the cardiac cycle that originates from the ventricle, characterized by aberration of the QRS complex (prolonged and distorted) and the T wave."

The PVC can be initiated at different moments of the electrical cardiac cycle: before the P wave (early ventricular premature complex), at the same time as the P wave (intermediary ventricular premature complex), after the P wave (late ventricular premature complex), or after a ventricular depolarization has already been initiated (fusion beat) [60].

The pacemaker site, which is the place where the electrical impulses are generated to initiate the depolarization, (and hence the contraction) is generally not located in the atrioventricular node as described in Section 2.3.1.B for the normal electric cycle. It could be positioned somewhere else in the ventricles particularly in the bundle branches, Purkinje network or ventricular myocardium. Because the depolarization does not follow the same sequence as in the normal case (starting by where it is generated), the shape of the QRS complex recorded in the electrocardiogram is often distorted and bizarre. The duration of the corresponding QRS is equal to or greater than 120 ms.

The interval between the peak of the PVC's R wave and the previous R wave's peak (the so called R-R coupling interval) is usually shorter than the rhythm considered normal for a particular patient. If the PVC does not depolarize the SA node, then the next P wave occurs at the expected time and (if the next beat is normal) the R-R interval to the next QRS is larger than the underlying rhythm.

In summary, the shape of the QRS for a PVC varies, depending on factors such as:

- what point of the cycle it occurs
- where the pacemaker site is
- how many pacemaker sites there are
- whether or not the PVC depolarizes the SA node.

It should be noted that many shapes of QRS complexes can designate the presence of PVC. For example, if the PVC is originated from the left ventricle, its QRS complex can resemble that of a right branch block (refer to Section 2.3.2.B.b). Similarly, if the PVC originated from the right ventricle, its QRS complex can resemble that of a left branch block (refer to Section 2.3.2.B.c). Moreover, a PVC starting from a bundle branch may appear only slightly bizarre or even look normal if generated near the bifurcation of the bundle of His [33]. Details on the electrical characteristics of this kind of beats may be reviewed in [7], [25], [33], [60], and [79].

b. Right Bundle Branch Block

According to [33], it consists in a "defective conduction of electrical impulses through the right […] bundle of His to the Purkinje […]. It may be complete or incomplete or permanent or intermittent". Note that the impulse conduction is not obstructed nor delayed through the unaffected conducting fibers, in this case the left bundle branch.

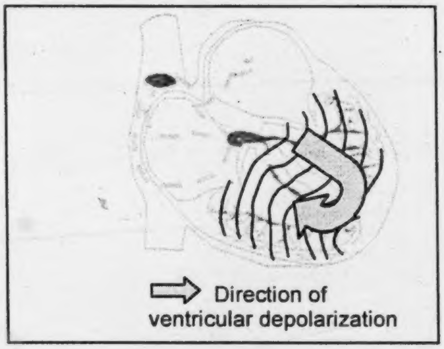


Figure 2-8: Depolarization sequence when there is a right bundle branch block [33]

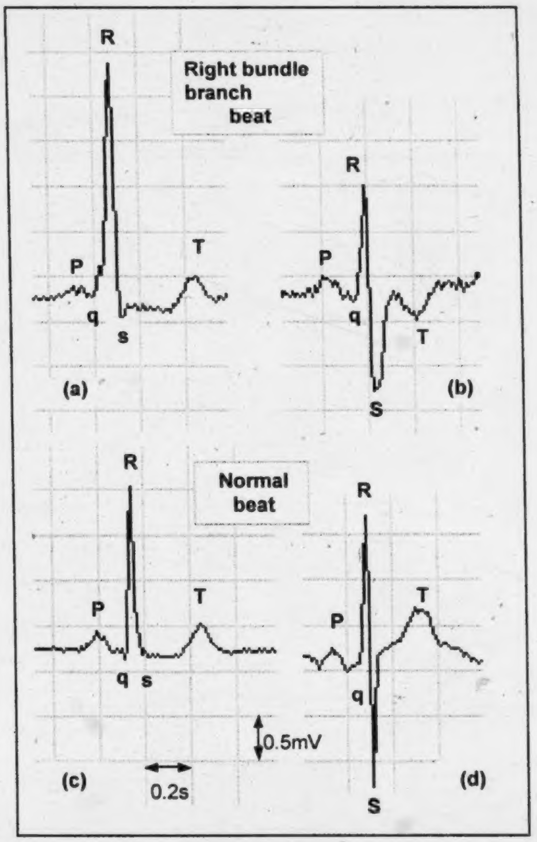


Figure 2-9: Example of right bundle branch block beats (RBBB), compared to normal beats (N)

The components are depicted as they appear in lead II. (a) and (b) are examples of RBBB, where the q wave does not differ particularly from the N beats and the R wave is prolonged and tall. Note that the S wave can be either deep or practically inexistent. The T wave can be positive as in (a), relatively small or negative as in (b). The normal beats are presented for comparison purposes. Note the differences in the duration of the different components of the QRS complexes.

Figure 2-8 shows the direction the depolarization takes and Figure 2-9 presents examples of beats obtained with right bundle branch block compared to normal beats. The depolarization process in the ventricles begins normally, hence producing a normal Q deflection in the electrocardiogram. The impulses then reach the bifurcation from where they are directed through the left bundle branch but do not go through the right side. This absence of stimulation from the right bundle branch produces a left side depolarization, which depending on the position of the heart may produce a taller and broader R deflection midway in the QRS development. The terminal depolarization phase occurs when the right ventricular wall slowly receives the depolarization wave from apex to base starting from the left ventricle. This causes an increment of the QRS overall duration and a radical alteration of the final deflection towards the right base (deep S wave) because the left ventricle is already totally depolarized. Because the depolarization process of the right ventricles is altered, the repolarization sequence is affected, producing smaller or even inverted T waves. If the block is not complete, the resulting recording in the electrocardiogram could easily be confused with the one of a normal beat.

c. Left Bundle Branch Block

It is written in [33] that the left bundle branch block (LBBB) consists of a "defective conduction of electrical impulses through the [...] left bundle of His to the Purkinje [...]. It may be complete or incomplete or permanent or intermittent". The impulse conduction is not delayed in the unaffected fibers of the right bundle branch.

Although it could eventually show without evidence of heart disease, the pattern of left bundle branch block is generally associated with heart disease in more cases than it is the right bundle branch block [25].

A basic representation of the depolarization sequence in the case of the left bundle branch block is shown in Figure 2-10. The impulse conduction goes unperturbed through the right conduct and then passes to the left ventricle through the intra ventricular septum provoking a slow depolarization on the left ventricle from right to left. This depolarization can travel different ways depending on where the left bundle is blocked and where it still conducts past the obstruction.

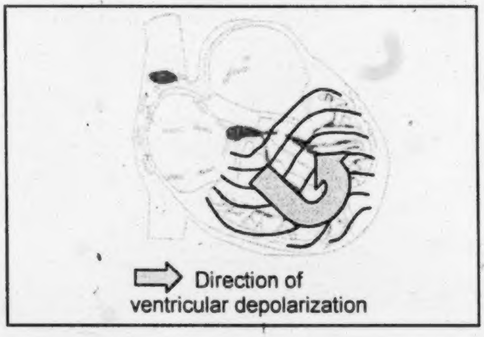


Figure 2-10: Depolarization sequence when there is a left bundle branch block [33]

Figure 2-11 shows two examples of the readings obtained in the electrocardiogram from lead II for both left bundle branch block beats and normal.

With the loss of conduction at the common left bundle branch, the intra ventricular septum becomes activated from the right bundle branch, resulting in a delayed depolarization of the left ventricle and a subsequent abnormal QRS complex, as shown in Figure 2-11(a) and

Figure 2-11(b). The septum undergoes a depolarization from right to left, contrary to the normal direction. The initial portion of the QRS complex in the left bundle branch block is drastically altered. The Q wave is not visible in either of the LBBB examples. The Q wave can be obliterated and replaced by a slowly developed, slurred R wave as in Figure 2-11(a).

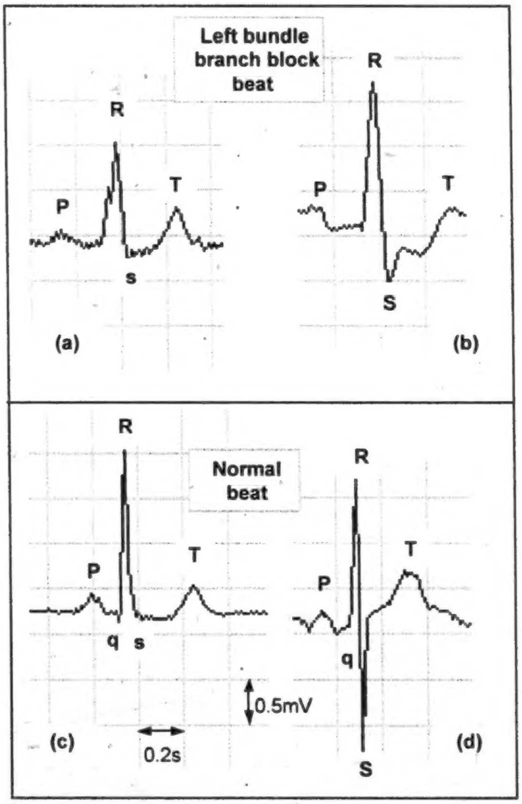


Figure 2-11: Example of left bundle branch block beats (LBBB), compared to normal beats (N).

(a) Example of a recording of a LBBB where the initial depolarization of the right ventricle unopposed by the left side eliminates completely the Q wave and starts a slurred R wave. This is followed by the depolarization of the left ventricle, which heterogeneity produces a notched appearance. The recording in (b) is another example of LBBB. In this case the Q wave stills absent, but instead of a notched line, the R wave is wide and relatively tall followed by a relatively deep S wave. This effect could be caused by the impulses being conducted by the bundle branches beyond the blocked Section. Note that the R wave is wider than in the case of the RBBB and the S wave is not as deep and wide than in the RBBB.

During mid-QRS, it is the activity in the right ventricle what is recorded. In effect, the depolarization in the right side is initiated by the normal conduction system to that chamber. This depolarization is unopposed by the left ventricle, which contributes with a relatively small and brief electrical force to this mid portion of the QRS complex. After passing though the right side, the depolarization wave reaches the left ventricle. The depolarization is delayed with respect to the one in the right side, because it is initiated only when the wave in the right ventricle reaches the cells adjacent to the left ventricle. The depolarization goes from the lower end of the intra ventricular septum or the apex of the right ventricle toward the base of the left ventricle. It can cause an upright wide and often "feathered" deflection in leads having left sided electrodes, as in Figure 2-11(b), but the heterogeneity of depolarization can also give the notched appearance to the resultant deflection as in Figure 2-11(a).

In all LBBB's, the QRS complex appears bizarre (abnormal in size and shape), and its duration is greater than 120ms. In certain cases, in which the wave front may enter the fibers beyond the conduction block, the QRS duration may be as long as 180 ms. The QRS can then be confused with the one in ventricular arrhythmias (wide, bizarre, and even notched). The polarity of the QRS complexes in lead II depends on the mean frontal plane QRS axis.

If the bundle branch block is partial or incomplete, the delay of the depolarization is lesser. Consequently the QRS complex is greater than 100ms but less than 120ms in duration, and often appears normal (narrow and sharply pointed).

C. Common causes for distortions of the readings

The readings on the electrocardiogram for lead II have been described above for normal beats and for three of the numerous abnormal classes of beats. These descriptions pertain to those conditions when the signal is not distorted by external factors. What follows is a brief review of the most common causes for distortions of the readings.

Abnormal waves and spikes in an ECG that result from sources other than the electrical activity of the heart and interfere or distort the components of the ECG are called artifacts. It is said in [33] that "the causes of artifacts are muscle tremor, alternating current (AC) interference, loose electrodes, biotelemetry-related interference, and external chest compression".

For example, muscular motions by the patient caused by nervousness, anxiety or even a medical condition are responsible for small oscillations in the baseline as shown in Figure 2-12 [60]. In the figure, the rapid and quasi-regular tremor is associated with Parkinson disease. How even these oscillations are depends on how regular the tremors are.

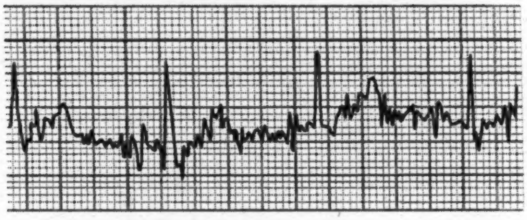


Figure 2-12: Example of the effects of muscle tremor on the recordings [33]

Because the recordings depend highly on the relative position between the heart and the electrodes, when this relative position is altered (by moving the part of the body where this electrode is, for example) the value of the isoelectric point changes causing the baseline to drift. This effect is shown in Figure 2-13 [60], where the record illustrates how the baseline is deviated when the patient raises the right arm where the RA electrode is attached.

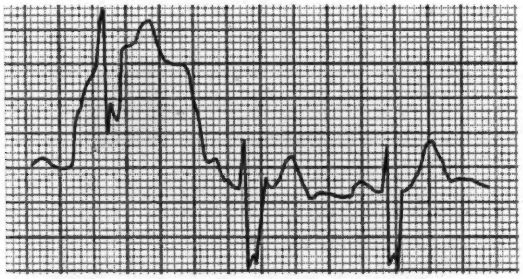


Figure 2-13: Example of the effect of patient movement [60]
In this case the baseline is wandering because of a movement of the right arm on which was tapped the RA electrode.

AC interference can be registered in several cases as, for example, a deficient connection on one ground electrode, the use of a poorly grounded AC operated ECG machine, or even the influence of neighboring high-tension wires or transformers. Figure 2-14 [60] depicts a 60-cycle ground interference produced by the RL electrode not being connected.

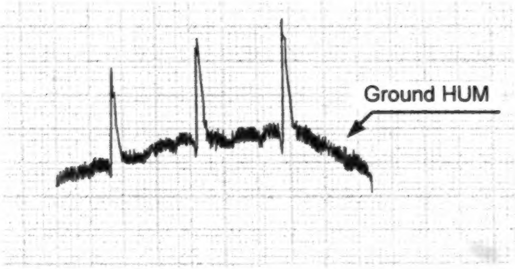


Figure 2-14: Effect of a 60 cycle ground interference on the ECG recordings [60]
The electric recordings of this example correspond to lead I. The characteristic waves lie on top of an oscillating baseline with a low frequency component (due to the 60-cycle electrical interference) and high frequency elements that result from irregular electrical currents no longer eliminated by the ground lead

A sub-optimal record possibly will be obtained if there is no good contact between the skin and the electrode. In effect, the resulting reading can present multiple sharp spikes and waves as shown in Figure 2-15 [33].

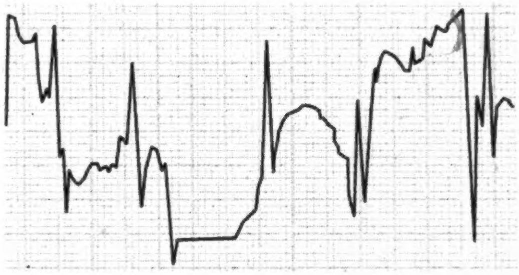


Figure 2-15: Example of the effect of loose electrodes on the ECG recordings [33]
Poor contact with the skin can cause multiple sharp spikes and unidentifiable waves in the ECG.

To "automatically" reduce the effect of these distortions on the electrocardiogram, some filtering of the signal should be done (refer to Section 2.5).

2.4. Mathematical background

In this Section, some basic concepts used in this thesis are explained. The first concept presented in Section 2.4.1, is the linear regression, which is used in this work to reduce the influence of the linear component of the signal during the modeling process. A short review of the normalization concept and techniques is offered next in Section 2.4.2. Entropy is introduced in Section 2.4.3 and its importance for clustering-based classification is presented. Finally, different measures of performance of classifiers are described in Section 2.4.4, to make clear how the results of this work are obtained and what they mean.

2.4.1. Linear regression

Regression is used in the analysis of models as a technique to express the relationship between several independent or predictor variables and a dependent or criterion variable [74].

This relationship is modeled by a regression curve of the dependent variable on the independent variables that most nearly fits the data (i.e. that best predicts the dependent variable from the independent ones). If the chosen regression curve is a straight line, then the regression is referred as linear regression [73]. The optimal equation of the line is found using the least squares method, by which the squared deviations of the observed points from that line are minimized.

The case of the electrical recordings of the heart is treated as a time series:

Given a set of d points of coordinates (t_i, s_i) for $i=1,...,d$, where s_i is the voltage recorded in the electrocardiogram from one of the leads at time t_i and d is the number of samples recorded for that segment.

The straight line that fits the data in the interval $[\, t_1, t_d\,]$ has the form:

$$r_i = a \cdot t_i + b \tag{2-1}$$

The parameters a and b of the linear regression equation are the result of solving a MSE problem [68],[73], that is:

- The positive deviations of the scattered points above the line cancel out their negative deviations of the scattered points below the line i.e. $\sum_{i=1}^{d} (s_i - r_i) = 0$.
- The sum of the squared differences between the scattered points s_i and r_i are minimum; i.e. $\sum_{i=1}^{d} (s_i - r_i)^2$ is minimized with respect to a and b.

The value of a and b in equation (2-1) are computed using the following formulas [73]:

$$b = \frac{d\left(\sum_{i=1}^{d} t_i s_i\right) - \left(\sum_{i=1}^{d} t_i\right)\left(\sum_{i=1}^{d} s_i\right)}{d\left(\sum_{i=1}^{d} t_i^2\right) - \left(\sum_{i=1}^{d} t_i\right)^2} \tag{2-2}$$

$$a = \frac{\sum_{i=1}^{d} s_i - b\sum_{i=1}^{d} t_i}{d}$$

(2-3)

2.4.2. Normalization

The contents of this Section is based on the information presented in [6], [58], and [16].
Given a data set **X** formed by N d-dimensional vectors, where each vector is formed by d attributes or features, the normalization of **X** consists in changing the range and/or distribution of the values of each feature in the mentioned data set. The normalization is applied to each variable separately.

In this Section three types of normalization are described:

- normalization using the range method
- the Z-score normalization
- the logistic normalization

Figure 2-16 shows an example in which the data set has 4 features and 10 samples. It can be seen in both Table 2-2 and Figure 2-16 that each attribute takes its values in different ranges and distributes its values differently. The figure shows a close up of the distribution of the third feature. Note that both the upper two points (segment A) and the lower two points (segment B) are separated by 10 units.

Table 2-2: Values that each feature can take on the example (Data set consisting of ten 4-dimensional vectors).

Data #	Dim 1	Dim 2	Dim 3	Dim 4
1	100	2	20	500
2	99	1.9	10	400
3	98	1.5	-178.7	300
4	20	1.2	-180	200
5	21	1.1	-183	100
6	5	1	-184	0
7	1	0.9	-185	-100
8	0.6	0.8	-187	-200
9	0.5	0.7	-188	-300
10	0.4	0.6	-198	-400

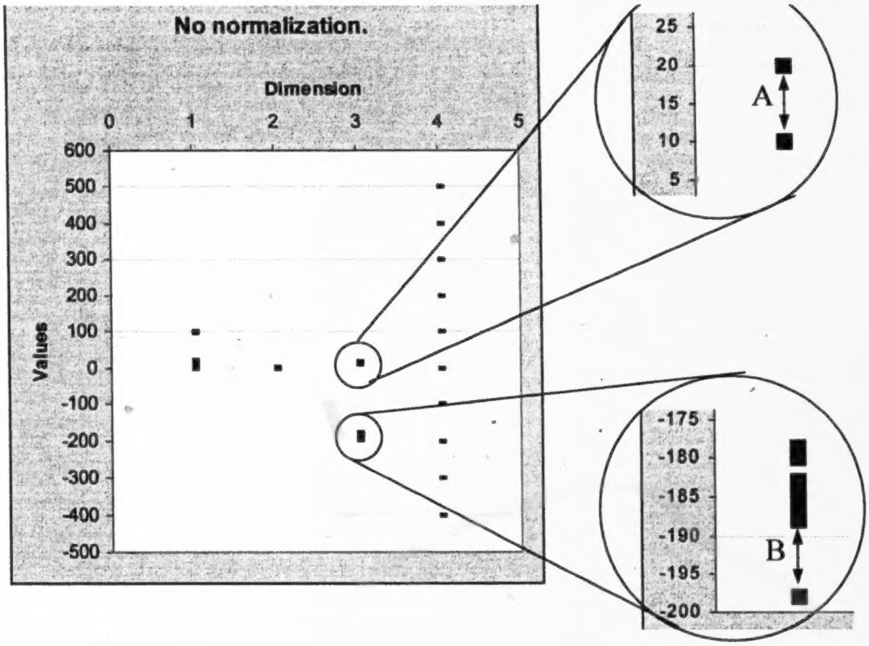


Figure 2-16: Example of non-normalized values in each feature for a data set containing ten 4-dimensional vectors.

There are several ways to normalize the data:

If only the range of the variables of the data set needs to be equalized, then each attribute is processed separately using the following formula:

Given,

$\mathbf{X} = \{x^i_j\}$ for i=1, 2, …, d and j=1,2 … N, the matrix of d column vectors $\{\mathbf{x}^i\}$ containing N elements.

$x^i_{min} = \min(\mathbf{x}^i), i = 1,2...d$

$x^i_{max} = \max(\mathbf{x}^i), i = 1,2...d$

$$\{x^i_j\}_{scaled} = \frac{x^i_j - x^i_{min}}{x^i_{max} - x^i_{min}} \quad (2\text{-}4)$$

This normalization is called the range method. It reduces the value of all the features to the range [0,1].

When applying this method in the example, the range of values is changed to [0,1] and the relationship among values of the same features is maintained as shown in Figure 2-17 and Table 2-3:

- The range of values of the fourth feature is now comparable to the three others. Note especially how the values along variable 2 are now very similar to values along the fourth variable.
- Any computation on the normalized data set is now equally affected by all the variables because they are all in the same range.

Table 2-3: Scaled version of the example data set

Data #	Scaled Variable 1	Scaled Variable 2	Scaled Variable 3	Scaled Variable 4
1	1	1	1	1
2	0.990	0.929	0.954	0.889
3	0.980	0.643	0.089	0.778
4	0.197	0.429	0.083	0.667
5	0.207	0.357	0.069	0.556
6	0.046	0.286	0.064	0.444
7	0.006	0.214	0.060	0.333
8	0.002	0.143	0.050	0.222
9	0.001	0.071	0.046	0.111
10	0	0	0	0

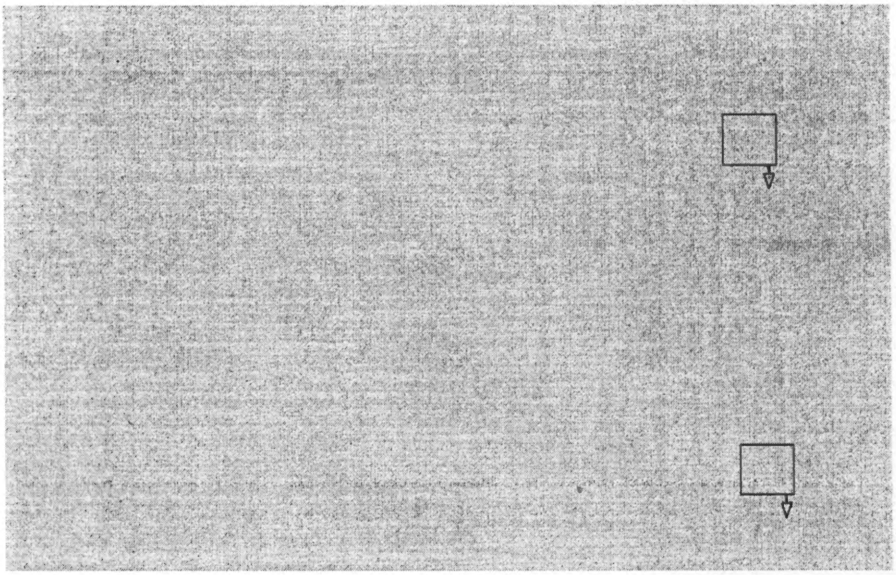


Figure 2-17: Example of normalized values in each variable for a data set containing 10 vectors, each with 4 features, using the range method of normalization.

The z-score normalization equalizes the spread for all the variables by making each of them have zero mean and unit standard deviation. The formula for normalizing the i th feature coordinate is:

$$\left\{x_j^i\right\}_{z\text{-score}} = \frac{x_j^i - \mu^i}{\sigma^i} \tag{2-5}$$

where μ^i and σ^i are respectively the mean and standard deviation of all values along each vector $\mathbf{x}^i$, for i=1, 2, ..., d.

When applying this method in the example, the origin of the values is shifted to zero and the standard deviation for all the variables is the unity, which makes the range of all the variables comparable without being strictly in [0,1]. This can be appreciated qualitatively in Figure 2-18.

Table 2-4: z-score version of the example data set

Data #	z-score variable 1	z-score variable 2	z-score variable 3	z-score variable 4
1	1.450	1.711	1.952	1.486
2	1.428	1.504	1.834	1.156
3	1.406	0.680	-0.393	0.826
4	-0.322	0.062	-0.409	0.495
5	-0.300	-0.144	-0.444	0.165
6	-0.655	-0.350	-0.456	-0.165
7	-0.743	-0.556	-0.468	-0.495
8	-0.752	-0.762	-0.491	-0.826
9	-0.754	-0.968	-0.503	-1.156
10	-0.757	-1.174	-0.621	-1.486

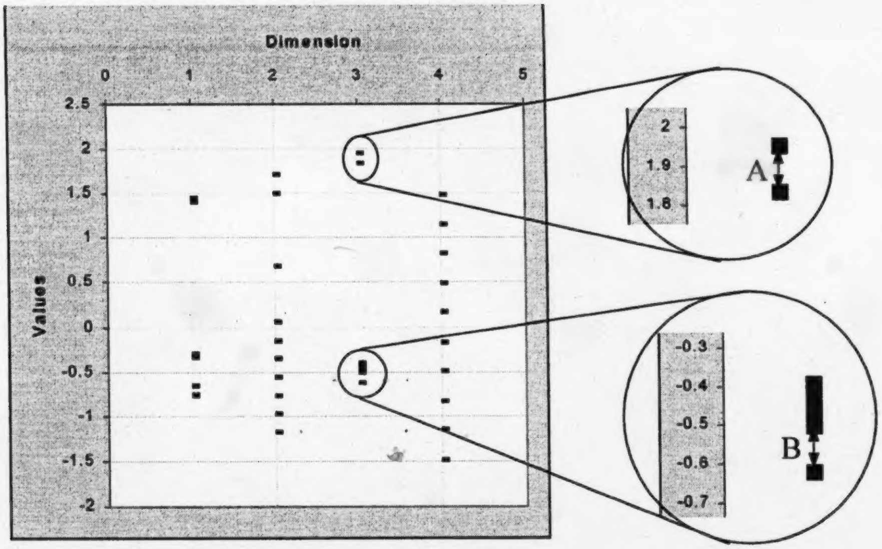


Figure 2-18: Example of normalized values for each one of the 4 features of the data set containing 10 samples, using the z-score normalization.

The logistic normalization is a non-linear scaling of the z-score. In effect, the formula for normalizing the i $^{\text{th}}$ feature coordinate is:

$$\left\{x_j^i\right\}_{\text{logistic}} = \frac{1}{1+\exp\left(-\left\{x_j^i\right\}_{\text{z-score}}\right)} \tag{2-6}$$

This type of normalization is very similar to the z-score around the mean values but it condenses the resolution of greater values. Figure 2-19 shows how this type of normalization affects the data set of the example. Note that on the third feature segment A is now smaller than segment B because the latter is closer to the mean value (i.e. closer to zero when using z-score).

Table 2-5: logistic version of the example data set

Data #	logistic Dim 1	logistic Dim 2	logistic Dim 3	logistic Dim 4
1	0.810	0.847	0.876	0.816
2	0.807	0.818	0.862	0.760
3	0.803	0.664	0.403	0.695
4	0.420	0.515	0.399	0.621
5	0.426	0.464	0.391	0.541
6	0.342	0.413	0.388	0.459
7	0.322	0.364	0.385	0.379
8	0.320	0.318	0.380	0.305
9	0.320	0.275	0.377	0.239
10	0.319	0.236	0.350	0.184

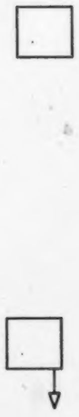


Figure 2-19: Example of normalized values for each one of the 4 variables of the data set containing 10 samples, using the logistic normalization.

Finding the type of normalization that provides the best representation of the structure of the data often requires following a trial and error process.

2.4.3. Entropy

In physics, the entropy provides a "measure of the disorganization or degradation of the universe, resulting in a decrease in available energy" [78].

To adjust the concept of entropy to the context of this thesis, an experiment involving a group of N patterns **X**=[$\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_N$] that are labeled as pertaining to any of c possible classes **class**=[class $_1$, class $_2$, …, class $_c$] may be considered. Given N_i, the number of patterns in **X** pertaining to class$_i$, the entropy H is defined as ([6], [59]):

$$H = -\sum_{i=1}^{c}\left(\frac{N_i}{N}\right)\log_2\left(\frac{N_i}{N}\right) \tag{2-7}$$

The entropy then gives a notion of the heterogeneity of the data. It takes its maximum value when all the classes are equally probable; that is when

$$p_i = \left(\frac{N_i}{N}\right) = \frac{1}{c} \quad ,\forall i = 1, 2, \ldots, c \tag{2-8}$$

The smallest value of entropy occurs when all the samples belong to a single class.

An example to illustrate this concept is shown in Figure 2-20, where the probabilities of each class are presented as percentages. There are four possible classes. Each horizontal bar corresponds to one possible distribution of the classes; the corresponding entropy is shown on the vertical axis. The bars are organized from top to bottom so that the value of their entropy can be visualized in ascending order. It can be observed in Figure 2-20 that the entropy increases when the probabilities of the classes are more alike.

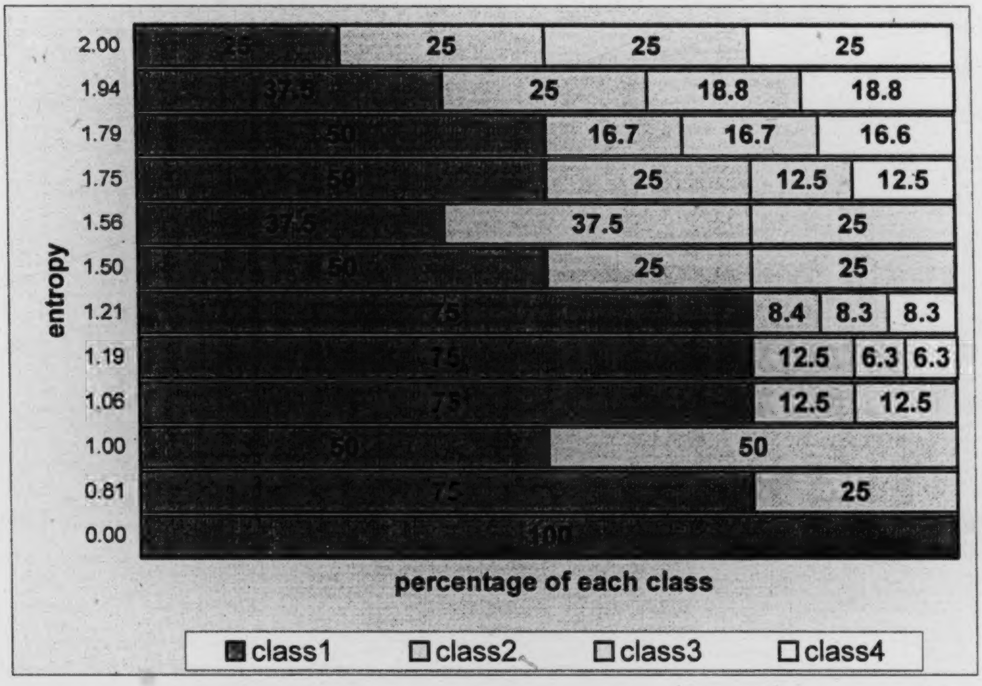


Figure 2-20: Example using four classes: Distribution of the classes vs. entropy.

In this work, the entropy is a criterion used to determine how suitable a group of samples is to represent the dominant class in it. In this case, the objective is for this group of samples to have the smallest possible entropy.

2.4.4. Measures of performance

A. Accuracy / Error

The simplest way to evaluate the performance of a classification process is computing either its accuracy or its error.

The accuracy of a classifier indicates the ratio of patterns that are classified correctly with respect to the total number of patterns that were presented; that is:

$$\text{Acc} = \frac{N_{\text{correct}}}{N} \qquad (2\text{-}9)$$

where N_{correct} is the total number of patterns that were correctly classified into c possible classes and N is the total number of patterns presented to the classifier.

Note that this general measure does not provide any information on the number of patterns correctly classified per class.

Similarly, the error is the ratio of patterns that are incorrectly classified with respect to the total number of patterns presented to the classifier; that is:

$$\text{Error} = \frac{N_{\text{incorrect}}}{N} = \frac{N - N_{\text{correct}}}{N} = 1 - \text{Acc} \qquad (2\text{-}10)$$

where $N_{\text{incorrect}}$ is the number of patterns that where not assigned to the correct class.

Note that like the accuracy, the error gives a general measure of misclassification but it does not give any idea of the distribution of all the possible misclassification combinations.

Both measures can be expressed as percentages:

% Acc = Acc * 100

% Error = Error * 100

B. Confusion Matrix

Given [Class_1, Class_2, …, Class_c] the c possible classes, the confusion matrix is the c x c matrix in the form:

Classified As→ Real Class ↓	Class_1	Class_2	…	Class_i	…	Class_j	…	Class_c
Class_1	C_{11}	C_{12}	…	C_{1i}	…	C_{1j}	…	C_{1c}
Class_2	C_{21}	C_{22}	…	C_{2i}	…	C_{2j}	…	C_{2c}
…	…	…	…	…	…	…	…	…
Class_i	C_{i1}	C_{i2}	…	C_{ii}	…	C_{ij}	…	C_{ic}
…	…	…	…	…	…	…	…	…
Class_j	C_{j1}	C_{j2}	…	C_{ji}	…	C_{jj}	…	C_{jc}
…	…	…	…	…	…	…	…	…
Class_c	C_{c1}	C_{c2}	…	C_{ci}	…	C_{cj}	…	C_{cc}

The confusion matrix gives specific information about the correctly classified patterns per class in its diagonal elements [C_{11}, C_{22}, …, C_{ii}, …, C_{jj}, C_{cc}].

The other elements of the matrix provide details on how the patterns were misclassified. For example the element C_{ij} is the number of patterns pertaining to class i that were classified as pertaining to class j.

The total number of patterns that were classified is the sum of all the elements of the matrix; that is $N = \sum_{i=1}^{c} \sum_{j=1}^{c} C_{ij}$

C. Kappa Score

The accuracy of a classifier was defined above as the percentage of patterns that are classified correctly. When the number of patterns of each class is different, this percentage depends not only on the performance of the classifier but also on a factor called concordance by chance. In effect, a classifier could be mediocre but still present high classification accuracy by assigning, for example, most of the patterns to the dominant class. The Kappa score is a measure of performance that clears the result of any coincidental agreement [19]. The kappa score maximum value is equal to one showing that all the patterns are classified correctly. When the kappa score is zero, any apparent correspondence in the classes is coincidental.

The kappa score is calculated using the following formula:

$$\text{KappaScore} = (P_o - P_c)/(1 - P_c) \tag{2-11}$$

where:

- P_o is the observed performance equivalent to the accuracy measure of equation **(2-9)** that can be obtained from the diagonal elements of the confusion matrix using:

  $$P_o = \left(\sum_{i=1}^{c} C_{ii}\right) \Big/ N \tag{2-12}$$

  where c is the number of classes, N is the total number of patterns and C_{ii} is the i^{th} diagonal element of the confusion matrix.

- P_c is the coincidental performance obtained by using the formula:

  $$P_c = \frac{\sum_{i=1}^{c}\left[\left(\sum_{j=1}^{c} C_{ij}\right)\left(\sum_{j=1}^{c} C_{ji}\right)\right]}{N^2} \tag{2-13}$$

  Another way to see P_c is by the formula:

  $$P_c = \sum_{i=1}^{c}\left(p(TC_i)\, p(OC_i)\right) \tag{2-14}$$

  where

$$p(\mathrm{TC}_i) = \frac{\#\,\mathrm{TrueClass}_i}{N}$$

(2-15)

Corresponding to the number of patterns that really pertain to class i in the input data set, divided by the total number of patterns. It is, in other words, the probability of occurrence of class i in the input data set

$$p(\mathrm{OC}_i) = \frac{\#\,\mathrm{OutputClass}_i}{N},$$

(2-16)

corresponding to the number of patters that where classified as class i divided by the total number of patterns. That corresponds to the probability of occurrence of class i at the output of the classifier for the same input data set.

Table 2-6 shows a possible interpretation of the kappa scores values in terms of strength of agreement.

Table 2-6: Confidence of Kappa Scores [61]

Value of Kappa	Strength of agreement
<0.20	Poor
0.21-0.40	Fair
0.41-0.60	Moderate
0.61-0.80	Good
0.81-1.00	Very Good

To illustrate the concepts introduced in this Section, there is in Table 2-7 an example of a confusion matrix produced when classifying a data set of 58 patterns into 3 classes.

Table 2-7: Example of Confusion matrix

TC ↓ \ OC →	Class 1	Class 2	Class 3	Total TC
Class 1	100	9	10	(100+9+10)=119
Class 2	1	9	0	(1+9+0)= 10
Class 3	4	0	11	(4+0+11)= 15
Total OC	(100+1+4)=105	(9+9+0)=18	(10+0+11)=21	**144**

The accuracy is calculated following equation **(2-12)** as

$$\mathrm{Acc}=(100+9+11)/144 = 0.8333$$

The kappa-score is calculated using equations **(2-12)** and **(2-14)** as follows:

P_o=Acc=0.8333.

$$P_c = \left(\frac{105}{144}\cdot\frac{119}{144}\right)+\left(\frac{18}{144}\cdot\frac{10}{144}\right)+\left(\frac{21}{144}\cdot\frac{15}{144}\right) = 0.6264$$

$$\boxed{\text{KappaScore} = \frac{0.8333 - 0.6264}{1 - 0.6264} = 0.5537}$$

In the example, the accuracy implies a good performance of the classifier but the kappa score indicates the contrary. In fact, the calculations to obtain the kappa score not only take into account the 0.83 accuracy value but also the coincidental performance, which is quite high because most of the misclassified patterns were confused with the most probable class. Because class1 was the dominant class representing more than 70 percent of the data and because it was correctly classified, the deficient classification of the other classes did not have repercussions on the value of accuracy

2.4.5. Measures of fit

When selecting and extracting the significant features of the data to reduce the number of variables to process, it is important to have a criterion to determine how much the number of its variables is to be reduced.

In this sense, several methods to express how fit the transformation is can be used, depending on what is to be done with this data.

In this work four measures of fit are presented:

1. squared error
2. linear correlation coefficient or index.
3. linear Discriminant Analysis (LDA).

A. Squared Error

Given:

$\mathbf{u}_k = [u_{k,1} \ \ \dots \ \ u_{k,i} \ \ \dots \ \ u_{k,L}]$, the kth exemplar of the original signal containing L samples and $u_{k,i}$, its ith sample

$\mathbf{u}_{d,k} = [u_{d,k,1} \ \ \dots \ \ u_{d,k,i} \ \ \dots \ \ u_{d,k,L}]$, the approximation of $\mathbf{u}_k$ using d features and $u_{d,k,i}$, its ith sample.

$\varepsilon_k(d)$ = the measure of fit of the kth exemplar for a transformation into d features.

The squared error on an exemplar is the sum over all its samples of the squared differences between the value of the signal to approximate and the value of the signal approximated using d features.

In order words, the squared error is given by:

$$\varepsilon_k(d) = \sum_{i=1}^{L} (u_{k,i} - u_{d,k,i})^2$$

(2-17)

To obtain a unique value per class and per features number, the squared errors of each class are averaged for each value of d giving

$$E_c(d) = \left(\frac{1}{Kc} \sum_{k=1}^{Kc} \varepsilon_k(d) \right)$$

(2-18)

where,

Kc is the number of exemplars in class c, $E_c(d)$ is the average squared error for class c using d features for $d = \{d_{min}, \dots, d_{max}\}$.

If the maximum average squared error per class for all values of d is called $Emax_c$, then the normalized squared error per class is

$$Enorm_c(d) = \frac{E_c(d)}{Emax_c}$$

(2-19)

If the maximum average squared error over all classes is called $Emax$, then the normalized average squared error is

$$Enorm(d) = \sum_{\text{all classes}} \frac{E_c(d)}{Emax}$$

(2-20)

for $d = \{d_{min}, \dots, d_{max}\}$

The squared error is presented in this work as the average squared error per class $Enorm_c(d)$ and the average squared error over all classes $Enorm(d)$, for selected number of features $d = \{d_{min}, \dots, d_{max}\}$.

B. Linear Correlation coefficient or index

The linear correlation measures how well a linear equation describes the relationship between variables. In this case, this measure is useful because it shows the degree of relationship between the original signal and its approximation.

The coefficient of correlation gives a dimensionless value whose value increases as the degree of relationship does, with a maximum of 1 for a complete positive linear correlation, 0 for no correlation at all, and –1 for a complete negative linear correlation.

The linear correlation coefficient (r) is computed as:

$$r = \frac{n\sum_{i=1}^{N} x_i y_i - \left(\sum_{i=1}^{N} x_i\right)\left(\sum_{i=1}^{N} y_i\right)}{\sqrt{n\left(\sum_{i=1}^{N} x_i^2\right)\left(\sum_{i=1}^{N} x_i\right)^2}\sqrt{n\left(\sum_{i=1}^{N} y_i^2\right)\left(\sum_{i=1}^{N} y_i\right)^2}} \qquad (2\text{-}21)$$

where

- x , the variable representing the original signal.
- y , the variable representing the signal approximation.
- n , the number of paired samples of x and y .

C. Accuracy when applying LDA

One of the tasks performed in this work is to take a highly dimensional multi-class data set and model it to reduce the number of variables involved while maintaining or improving the separation among classes (see Section 2.6 for details concerning the data set).

The linear discriminant analysis (LDA) is explained in Section 3.2.The accuracy (concept explained in Section 2.4.4.A) of the LDA classifier is used in this work as a measure of fit to have a notion of how sparse the data is in the reduced space.

2.5. ECG automatic analysis

2.5.1. Elimination or reduction of the baseline wandering

The baseline of the ECG signal forms the voltage where the isoelectric level is. As mentioned in Section 2.3.2.C, there are several factors that may affect the baseline's characteristics. In this Section, some of the methods to reduce these factors are briefly described.

One of the causes of distortions of the ECG readings is the movement of the part of the body where an electrode is located. This causes among other things, abrupt fluctuations of the baseline. Because this event is cyclic but rather random a filter is not indicated. But because it is also almost inevitable that the patient will eventually move, ways to reducing its effect in the ECG must be considered.

Depending on the application, one might want to eliminate the fluctuation on the signal itself before studying it or eliminate its effects on the output of the application.

An example of the latter case is the method Okada presents in [55] to detect QRS complexes. In the mentioned algorithm, the output is supposed to be a positive pulse for each R peak. Because his algorithm generates a peak also when there was a stepwise fluctuation of the baseline, he proposed to eliminate the false peaks by setting to zero the peaks in the output that do not occur when the input signal is not at a local maximum or minimum. The reason for it is that the Q wave, the R wave, and the S wave have all some symmetry on the time axis. The designer has, of course, to establish the size of the window to take into account this local minimum or maximum, preferably corresponding to the average duration of each component of the QRS complex. He proposes 28 ms, but then, this parameter may vary from one patient to another, or from one period of time to another, from one class of beats to another.

An example where the objective is to eliminate the fluctuation of the baseline directly on the ECG recording is the work presented by Rasiah and Attikiouzel [65]. Before proceeding with the extraction of features by modeling the ECG into Hermite basis functions, the authors divide the signal into windows in each of which they fit a linear equation through the samples. This fitting is done through linear regression as explained in Section 2.4.1. The idea behind this procedure is that, when it is applied on an adequate window size, the shape of the beats would have approximately the same orientation and translation unless the beat itself is different. Once again a parameter has to be adjusted, which is the size of the window.

A rather straightforward method is presented in [67] to define this "reference line", in which the derivation of some features is based. The authors take the first and last point of the window and generate the line from there. They then proceed to generate the features as the difference from the signal to the line created and its derivative. Before generating the features, it might be advisable to filter the to reduce higher frequencies likely to affect the appropriateness of the reference line.

In most of the respective literature [54], [65], [72], [82], [44], [43], [55], [77], [1], [57], [29], [2], [36], and [67] the authors mention the necessity and often explain in detail their method for filtering the signal using a band pass filter before any other procedure.

In brief, "The bandwidth of the band pass filter must be chosen to reflect the tradeoff between noise reduction and loss of high frequency details. If the bandwidth is too large, noise reduction suffers. If the bandwidth is too narrow, high frequency QRS characteristics are lost." [36]

In order to establish the frequency range in which the filter should be set, the duration of the QRS in seconds in the application has to be checked. A rough estimate of the extreme frequencies if only the QRS is needed to pass through the filter could be for the lower frequency, the inverse of the expected maximal duration of the QRS, and for the higher frequency, the inverse of the expected minimal duration of the QRS or of one cycle in it (as in notched QRS like in the LBBB case). In [2], the authors consider the range to be below 40 Hz. In [57], the band pass goes from 5Hz to 11Hz. In [55], the frequency of the low pass filter is approximately 35Hz.

With the band pass filters, noise such as the muscle tremor and AC interference do not (ideally) get through the low pass filter (set at most in 40Hz). The high pass filter above 5 Hz attenuates somewhat the waves formed by lower frequencies such as the P and the T wave, which facilitate the QRS detection.

2.5.2. Modeling the QRS

A. Introduction

In the literature, a number of attempts for determining the ideal number of features that describe a digital QRS signal are presented. In [11], the authors directly used raw data, trace segmentation, and polygonal approximation. In [63], the features were more refined: The authors used average heart rate, and energy relations between bands of frequency in the signal. In the work presented in [71], the authors extracted morphological information of the beats such as P-R segment duration, width of the QRS, prematurity degree, and others. Principal component analysis was used in [32] in conjunction with R-R interval, and width of the QRS complex. Several kinds of transformations are also presented in the literature, such as Discrete Wavelet Transforms (DWT) ([36], [14], [47], [12], [11], [15]), Discrete Fourier Transform (DFT) [15], Discrete Cosine Transforms (DCT) [3], and Hermite Series of Expansion Transform ([65], [66], [72], [82], [43]). Depending on the application, one transformation may give better results than others. In the classification case for example, the reduction of the number of variables of the data is sought as well as a good interclass separation to enhance the discrimination among classes in the feature space. In this sense, several of the studies mentioned above show that, in the case of the QRS segments, using Hermite basis functions on an orthogonal transformation satisfies this need.

B. Hermite series of expansion

The n^{th} order Hermite function is a weighted function of the n^{th} order Hermite polynomial. Given a continuous signal $u(t)$, it can be represented by an infinite series of Hermite basis functions:

$$u(t) = \sum_{i=0}^{\infty} \left(a_i \, h_i(t, \tau, \sigma) \right)$$

(2-22)

where

- a_i is the coefficient of the i^{th} basis function. This coefficient will be taken as a feature in the transformed space.
- $h_i(t, \tau, \sigma)$ is the i^{th} Hermite basis function defined as:

$$h_i(t, \tau, \sigma) = \frac{1}{\sigma} \exp\left(- \frac{(t-\tau)^2}{2\sigma^2} \right) \hat{H}_i\left(\frac{t-\tau}{\sigma} \right)$$

(2-23)

where

  - σ and τ are parameters to control the width and the shifting position of the curve respectively
  - $\hat{H}_i(t)$ is defined recursively:

$$\hat{H}_0(t) = \frac{1}{\sqrt{\sqrt{\pi}}}$$

$$\hat{H}_1(t) = t\sqrt{\frac{2}{i}}\,\hat{H}_{i-1}(t)$$

$$\hat{H}_i(t) = t\sqrt{\frac{2}{i}}\,\hat{H}_{i-1}(t) - \sqrt{\frac{i-1}{i}}\,\hat{H}_{i-2}(t)$$

(2-24)

In practice, only a finite number of terms are taken, and the infinite series is approximated to a series of expansion of n Hermite basis functions:

$$u_{n,\sigma,\tau}(t) = \sum_{i=0}^{n-1} \left(a_i\, h_i(t,\tau,\sigma)\right) \tag{2-25}$$

An example for a range of times between –0.5 to +0.5 seconds, width σ=0.1 and shift τ=0 of the first eight Hermite basis functions is shown in Figure 2-21.

Note the peculiar shape of each basis function. They all have a Gaussian-like shape, which makes them similar to the shape of the QRS recordings: the higher frequency concentrated toward the center of the window (for τ=0). How sharp these curves are depends on the width parameter σ. This fact is useful when describing the QRS waves, since they might be wide as in the LBBB and the PVC, not so wide as in the RBBB or relatively sharp as in the Normal beats. The first basis function (h_0) stands above the reference line and has one positive peak, similar to the positive QRS depicted in Figure 2-7. The second basis function (h_1) is biphasic and holds a certain similarity to the biphasic version of the QRS. The third basis function (h_2) has now a central negative peak and two positive waves on its side. This curve and the QRS with negative direction could be considered alike. It can be observed that for the next basis functions, the even functions (h_4, h_6...) will continue being symmetric with respect to the central vertical and with their central peak alternating between positive and negative. The odd functions (h_5, h_6...) also maintain a pattern: the right side inverted with respect to the left side of the vertical axis.

Because the ECG recordings are modeled a small sliding window at the time, the QRS are not always in the middle of it, that is why the shift parameter τ is included in the calculations.

There are n+1 peaks in the basis function h_n. For example h_7 holds 8 peaks in the same window where h_3 presents 4 peaks. This means that by modeling a QRS with more or less basis functions, one can control to some extent the frequency content of the modeled version.

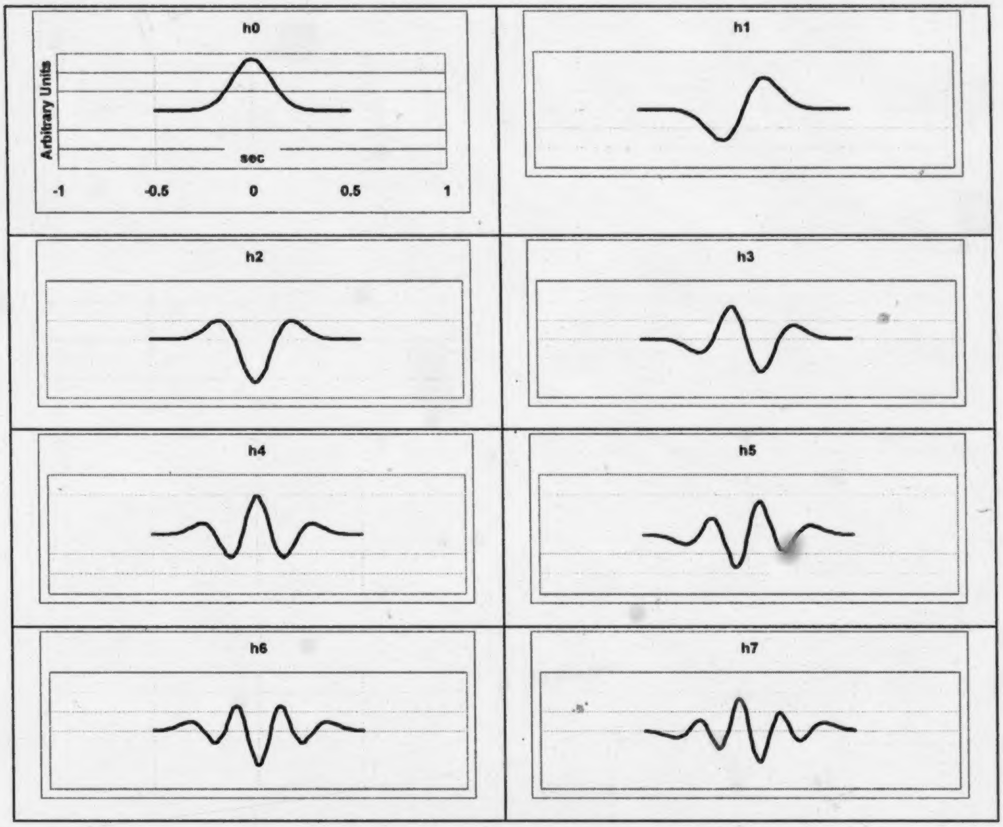


Figure 2-21: Plots of the first eight Hermite basis functions
($h_{n,\sigma}(t,\tau)$, n=0, …, 7), for t=-0.5s to t=+0.5s, σ=0.1, and τ=0.

Different techniques have been presented in the literature to obtain the value of the coefficients a_i in equations **(2-14)** and **(2-22)**.

For constant values of σ and τ, the conventional way to determine the values of each of these coefficients is by using the following expression [66]:

$$a_i(\sigma,\tau) = \int_{-\infty}^{+\infty} u(t)\, h_{i,\sigma}(t,\tau)\, dt \qquad (2\text{-}26)$$

This integral can be solved by any quadrature like the Gauss-Jacobi integration theorem as shown in [66], or the adaptive recursive Newton-Cotes 8 panel rule as in the MATLAB toolbox, which is based on the theory given in [20].

It is important to note from equation **(2-26)** that the value of the coefficients a_i depends on the values of σ and τ. Equation **(2-26)** does not solve the problem of calculating these two parameters. Some calculations have then to be included to complete the solution.

Rasiah et al. [66] propose to estimate iteratively the width and shift values in two phases: a step search approach and a gradient descent approach. For each iteration m, the values of the coefficients are calculated following **(2-26)**.

Given the error sum function E(σ,τ) defined as the squared error between the original signal and the signal modeled using determined values of σ and τ, the algorithm is as follows:

Phase one consists in evaluating the error for a grid of values of σ and τ, and selecting the values for which the error was minimum to be the initial values for the second phase.

Phase two, the gradient descent approach, involves adjusting iteratively the parameters σ and τ, starting with the initial values obtained from the previous phase. For the m th iteration, the values are updated first assuming a constant value of τ :

$$\sigma_{m+1} = \sigma_m - \mu_1 \frac{\partial E(\sigma)}{\partial \sigma} \qquad (2\text{-}27)$$

and then assuming σ is constant:

$$\tau_{m+1} = \tau_m - \mu_2 \frac{\partial E(\tau)}{\partial \tau} \qquad (2\text{-}28)$$

In the case of the QRS analysis presented in this work, the signal is discrete. Then $u(t)$ is now denoted $\mathbf{u} = [u_1 \ \dots \ u_k \ \dots \ u_L]$.

It is assumed that

$$u(t) = \begin{cases} QRS(t), & t \in [-M,+M] \\ 0, & \text{otherwise} \end{cases} \qquad (2\text{-}29)$$

Then **u** is formed by L equally spaced QRS samples corresponding to times $\mathbf{t} = [t_1 \ \dots \ t_k \ \dots \ t_L]$.

Since equation **(2-26)** assumes continuity of the functions in it, it is necessary to use interpolation techniques while solving the integral. The type of interpolation used to make the discrete signal continuous depends on the compromise between its accuracy and its computation expense. Possible interpolation methods include the nearest neighbor interpolation, the linear interpolation, and the cubic spline interpolation.

Laguna et al present another approach to determine simultaneously the parameters of the Hermite transformation [44]. They present the Adaptive Hermite Model Estimation System (AHMES). This method is based on a multiple-input adaptive linear combiner that allows for an online estimation of these parameters to make them fit the needs of the approximation problem. The block diagram of AHMES is shown in Figure 2-22.

The system accepts one sample u_k of the signal **u** at the time. For the kth sample of the input signal the basis functions $\{\boldsymbol{\Phi}_i\}$, i=1,...,(n-1) are generated accordingly to the value of σ. The output y_k of the system is the linear combination of these basis functions weighted by a factor $\mathbf{w}_n$. The error signal e_k is the difference between the kth sample of the original signal and the model output y_k. Namely,

$$e_k = u_k - y_k \tag{2-30}$$

This error is used to adjust the weights and width as for example in [82], where its derivative with respect to each variable for an iteration k is scaled by a factor μ called learning rate and added to the value of the variable in k to obtain the new value to use in (k+1) as shown in equations **(2-31)** and **(2-32)**.

$$w_{i(k+1)} = w_{ik} + \mu_1 \frac{\partial\left((e_k)^2\right)}{\partial w_i}, i = 0,\ldots, n-1 \tag{2-31}$$

$$\sigma_{k+1} = \sigma_k + \mu_2 \frac{\partial\left((e_k)^2\right)}{\partial \sigma} \tag{2-32}$$

where,

σ_k= width parameter on the kth iteration

w_{ik}= weights that approximate the coefficients of the Hermite series on the kth iteration

e_k = difference between the generated signal y_k and the original u_k.

The weights and width are adjusted for each iteration k. One epoch starts with the first sample of a QRS segment (k=1) and ends when one QRS segment has passed through (k=L). The process involves several epochs to conveniently adjust the parameters. The number of epochs depends, among other factors, on the time available until next QRS segment and the value of the error e_k.

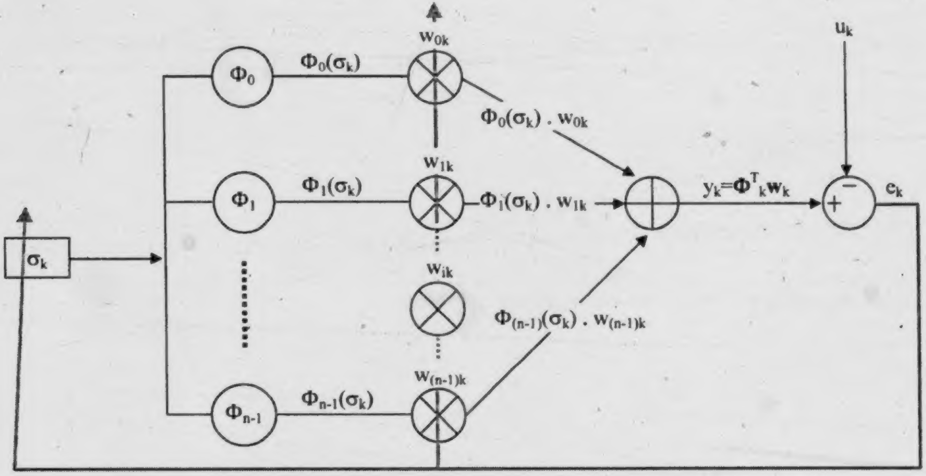


Figure 2-22: Block Diagram of AHMES [66].

C. Principal Component Analysis (PCA)

The purpose of feature extraction is to create a model of a group of data that contains fewer and hopefully more pertinent features than did the original representation. Factor analysis undertakes this goal [34]. The main objective in factor analysis is to reproduce as well as possible in the new model the correlation matrix of the original data, specially its variances. The Principal Component Analysis ([6], [16], [10]), also called Karhunen-Loeve or eigenvector transformation is a technique that embraces all of the above. It uses the statistic information of the given data set to project its features originally in an O-dimensional space to a d-dimensional space, where the d features are uncorrelated with each other and d is usually d smaller than O.

The axes in this new space are extremes (maxima and minima) and uncorrelated with each other, and weighted according to the amount of the total variance that they describe (Note that if the original variables are already uncorrelated, there is no point in performing principal component analysis).

Once the variance of each principal component and the total variance (which is the sum all these variances) are calculated, the fraction of variance of each component is computed as the ratio of its variance to the total variance.

When the main purpose is to decrease the number of variables of the data, the key to this analysis is to estimate the minimum value of d for which either of two conditions are satisfied:

- the fraction of variance of the d^th^ principal component is small enough.
- the cumulative variance for the d^th^ principal component is elevated enough.
- the value for the limits in the values of fraction of variance and cumulative variance has to be established by the designer.

The technique to obtain the principal components is explained as follows.

Given a matrix **X** of N O -dimensional patterns $\{\mathbf{x}_n\}$, n=1, 2, …, N, we want to transform it into **Y**, a matrix of N d-dimensional patterns $\{\mathbf{y}_n\}$, n=1, 2, …, N:

$$\mathbf{y}_n = \mathbf{A}\mathbf{x}_n \tag{2-33}$$

where matrix **A** defines the relationship between the original axes and the principal components. In other words, it is the transformation matrix for changing the coordinate system.

Because one of the premises is to maintain the variability from one space to the other, one must then have a look into the covariance of the data.

The covariance matrix $\mathbf{\Sigma}$ of the data set **X** in the original space is an O by O matrix obtained by [6]:

$$\mathbf{\Sigma} = \frac{1}{N-1}\sum_{n=1}^{N}\left(\mathbf{x}_n - \overline{\mathbf{x}}\right)\left(\mathbf{x}_n - \overline{\mathbf{x}}\right)^T \tag{2-34}$$

where

$\overline{\mathbf{x}}$ is the vector that contains the mean value of each variable of matrix **X**, given by

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \tag{2-35}$$

The covariance matrix of **Y** (i.e. the covariance matrix of the data in the new space) would be $\mathbf{\Lambda}$. It is related to the covariance matrix $\mathbf{\Sigma}$ in the original space by this equation [13]:

$$\mathbf{\Lambda} = \mathbf{A}\,\mathbf{\Sigma}\,\mathbf{A}^T \tag{2-36}$$

As previously mentioned, the principal components are uncorrelated; **Λ** is therefore a diagonal matrix, which diagonal values λ_i , for i=1, 2... O accounts for the variance of feature i in the transformed space.

Defining **W** as the transpose of **A**, equation **(2-36)** is written:

$$\mathbf{\Lambda} = \mathbf{W}^T \mathbf{\Sigma}\, \mathbf{W} \tag{2-37}$$

Defining **Z** as the correlation matrix of **X**, it is known that the correlation between two variables can be expressed as the covariance between them divided by the product of the standard deviation of each one[73]. Then, in terms of the present variables:

$$Z_{i,j} = \frac{\Sigma_{i,j}}{\sigma_i \sigma_j}, \text{ for } i,j = 1,2,\ldots,O \tag{2-38}$$

where

- $Z_{i,j}$ is the correlation between the i th variable and the j th variable.
- σ_i, σ_j are the standard deviation of the i th variable and the j th variable respectively.
- $\Sigma_{i,j}$ is the covariance between the i th variable and the j th variable.

If **X** has been normalized (to have unitary standard deviation) for all its features, then **Z**= **Σ** and equation **(2-37)** can be rewritten as:

$$\mathbf{\Lambda} = \mathbf{W}^T \mathbf{Z}\, \mathbf{W} \tag{2-39}$$

After some matrix manipulation on **(2-39)** and because **W** is orthogonal the following expression can be obtained:

$$\mathbf{W}^T \mathbf{Z} - \mathbf{\Lambda}\, \mathbf{W}^T = \mathbf{0} \tag{2-40}$$

where **0** is an O by O matrix of zeros.

This equation gives O sets of simultaneous equations, which for a non-zero solution for the elements of **W** to exist, must satisfy for each λ_i the so-called characteristic equation of the matrix **W**:

$$\left|\mathbf{\Sigma} - \lambda_i \mathbf{I}\right| = 0 \tag{2-41}$$

where

I is an O by O unit matrix

λ_i, for i=1, 2, …, O are called the eigenvalues of **Σ**

The columns of matrix **W** are the eigenvectors associated to each eigenvalues.

These eigenvectors $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_O\}$ are O by 1 vectors that satisfy the following equation:

$$(\mathbf{\Sigma} - \lambda_j)\mathbf{c}_j = \mathbf{0}$$

(2-42)

These vectors are orthonormal. Namely:

$$\mathbf{c}_j^T\mathbf{c}_k = 0 \text{ if } j \neq k$$

(2-43)

and

$$\mathbf{c}_j^T\mathbf{c}_k = 1 \text{ if } j = k$$

(2-44)

Because each eigenvalue is the share of the total variance of the corresponding component, the eigenvalues are sorted in descending order, and the corresponding columns of **W** (i.e. the eigenvectors) are re-arranged in the same way.

Finally, the first d columns of this "sorted" matrix are selected to form the final transformation matrix. The value of d is selected depending on the application, such that the sum of the d eigenvalues is greater than an acceptable value.

In summary, the ith O-dimensional pattern $\mathbf{x}^i \in \mathbf{X}$, has an image $\mathbf{y}^i \in \mathbf{Y}$, which is a d-dimensional vector. Equation **(2-33)** can then be rewritten as:

$$\begin{bmatrix} y_1^i & y_2^i & \ldots & y_d^i \end{bmatrix} = \begin{bmatrix} x_1^i & x_2^i & \ldots & x_O^i \end{bmatrix}\begin{bmatrix} \mathbf{c}_1 & \mathbf{c}_2 & \ldots & \mathbf{c}_d \end{bmatrix}, \text{ for } i=1, 2, \ldots, N$$

(2-45)

It has been shown above that PCA finds feature combinations that model the variance of a data set. These may not be the same features that separate the classes, i.e. the PCA components that model the largest contributions to the data set variance may not work as good as expected for pattern recognition. Taking into account this potential problem, the linear discriminant analysis (LDA) and its generalization for the multiple class problem, the multiple discriminant analysis (MDA) goes beyond principal component analysis: While

PCA seeks a projection that best represents the data in the least squares sense, MDA seeks a projection that best separates the data in a least squares sense.

LDA and MDA are explained in Section 3.2.

2.5.3. QRS Detection

A. Introduction

Even though the QRS detection problem is not the principal goal of this work, it is of interest to review briefly what has been done in this sense because it is necessary to first detect the QRS complexes in order to be able to classify them. There is a comprehensive review work by Köhler et al. [37], where many approaches to QRS detection are presented. Köhler et al define the structure of the QRS detection algorithm as divided into "[...] a preprocessing or feature extraction stage including linear and non linear filtering and a decision stage including peak detection and decision logic.", and the different algorithms they present are discriminated with respect to their preprocessing stages. The algorithms they present include approaches based on signal derivatives and digital filters, wavelet-based QRS detection, neural network approaches, adaptive filters, mathematical morphology, matched filters, Hilbert transform-based QRS detection, syntactic methods, and others.

In this Section, four algorithms are described. The first one, designed by Okada [55], is an early application of digital filtering techniques to the problem of QRS detection. The second algorithm is the one presented by Tompkins et al. [29] [57], which uses a processor based on band pass linear and non-linear filtering techniques with a decision stage for QRS detection. The third algorithm is the one proposed by S. Suppappola and Y. Sun in [77], [76]. In this algorithm only non-linear filtering is used. Finally, the fourth algorithm is an adaptation of the algorithm presented in [21], by which a model of the signal is constructed using neural networks, then trained to recognize when a QRS occurs.

Other attempts have been made for detecting QRS. Among them:

1) the use of discrete wavelet transforms [36][12][47], which offers promising results.
2) the use of the envelope (Hilbert transformation) of the ECG [54][43]: they applied this method on the MIT-BIH database and claimed a 99.7 % detection rate.

B. Okada's digital filter for QRS detection.

The digital filter presented by M. Okada [55] was developed to remove components other than the QRS from the electrocardiogram digital signal. The algorithm is depicted in Figure 2-23. The original signal, called y0 in Figure 2-23, goes through a band pass filter performed in two steps: First the signal is low passed to eliminate high frequencies (y1 in Figure 2-23) by a 3-point weighted moving average and then put through a stricter low pass filter (a $2m_1+1$ points moving average) to obtain the undesired low pass components (y1b). The two outputs of the low pass filters are then subtracted to obtain the band pass filtered version of the original signal. This result is then squared to make all differences positive (this corresponds to y2 in Figure 2-23). Because the QRS contains the higher frequencies in the remaining components of the signal, these higher frequencies are enhanced by multiplying the result of the previous step by a squared ($2m_2+1$ points) moving average of itself (obtaining y3). The next step exploits the characteristic of the Q, the R, and the S waves: they are (relatively) symmetrically shaped. The drifting in the baseline is usually not symmetrical and produces stepwise changes. Thus, in order to remove any spurious peaks, product of abrupt baseline wandering, the peaks with no symmetry in the time axis in the approximated duration of a QRS ($2m_s+1$) are eliminated (resulting in y4). Subsequently, the points below a certain threshold are set to zero, obtaining in this way what we will call the "signal of interest". An estimated of this threshold is proposed in Okada's work to be $(1/32)^{th}$ of the maximum value the last output (y4) takes in the 2m+1 samples long window around each point.

In his work, Okada estimated the most suitable value for m_1, m_2, m_s and m to be the same for all of them and equivalent to 12ms (3 samples @ 250Hz).

The points detected in less than 144 ms intervals are grouped as pertaining to the same segment with a characteristic initial point. In order to obtain a square wave, the segments of 144 ms duration, starting at the characteristic points obtained before, are set ON and all the rest OFF (y5 in the figure). Finally, the fiducial points of the QRS such as the peak of the R wave may be detected by searching the maximum amplitude of the first derivative in the interval obtained by the square wave of the "signal of interest".

Each step of the procedure presented in Figure 2-23 is illustrated in a practical example shown in Figure 2-24. The sample in this example was extracted from the MIT-BIH database that will be described later on this work (Section 2.6.1).

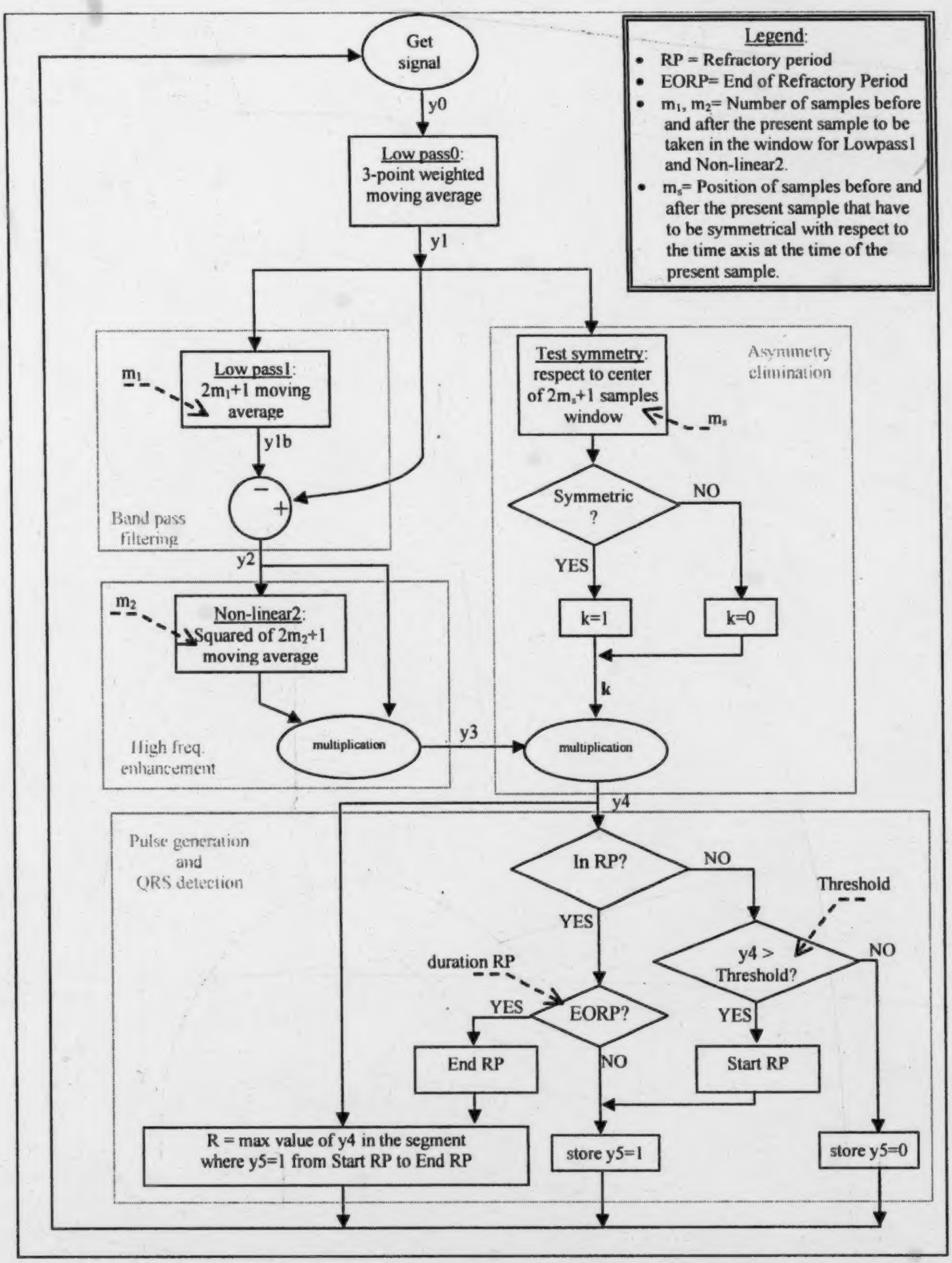
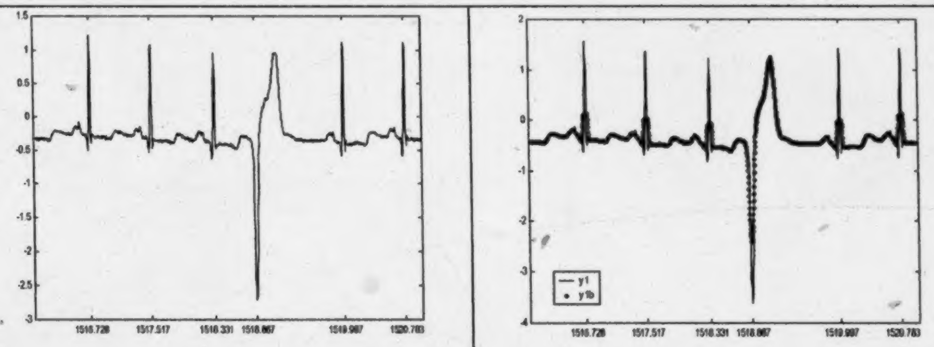

Get signal
y0
Low pass0: 3-point weighted moving average
y1
Legend:
RP = Refractory period
EORP= End of Refractory Period
m1, m2= Number of samples before and after the present sample to be taken in the window for Lowpass1 and Non-linear2.
ms= Position of samples before and after the present sample that have to be symmetrical with respect to the time axis at the time of the present sample.
m1
Low pass1: 2m1+1 moving average
y1b
−
+
Band pass filtering
y2
Test symmetry: respect to center of 2ms+1 samples window
ms
Asymmetry elimination
Symmetric ?
NO
YES
k=1
k=0
k
m2
Non-linear2: Squared of 2m2+1 moving average
High freq. enhancement
multiplication
y3
multiplication
y4
Pulse generation and QRS detection
In RP?
NO
YES
Threshold
y4 > Threshold?
NO
YES
duration RP
YES
EORP?
NO
End RP
Start RP
R = max value of y4 in the segment where y5=1 from Start RP to End RP
store y5=1
store y5=0

Figure 2-23: Flowchart of the algorithm proposed by Okada

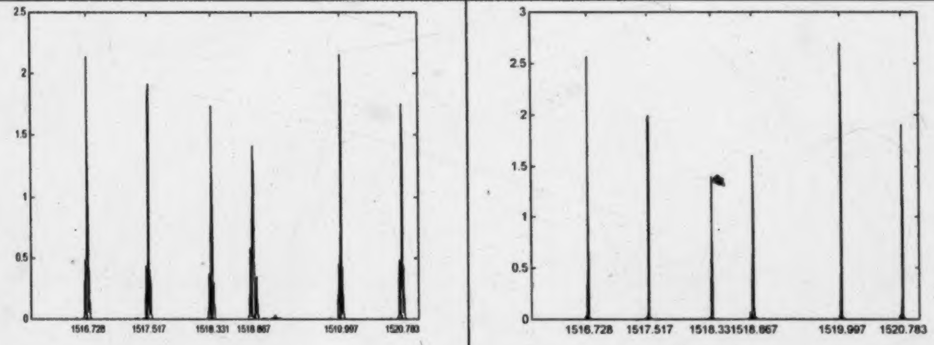
The original signal y0 goes through a low pass filter to eliminate higher frequency components. The resulting y1 goes through a stricter low pass filter to obtain y1b, the lower components of y1 that will be eliminated. y1b is subtracted from y1 to get y2, the band pass version of y0. y2 goes through a non-linear filter to enhance higher frequency components, obtaining y3. y4 maintains only the elements which time of occurrence is in the symmetry axis of the y1 signal in the 2m3+1 window. Each non-zero pulse indicating the position of the QRS starts when y4 exceeds the threshold. This pulse lasts the estimated duration of the refractory period and then goes to zero to wait for another starting point.

The original signal y0, sampled at 360Hz, contains six beats. These beats are shown in Figure 2-24(a). Five of these beats are normal (N) and one corresponds to a premature ventricular contraction (V). The band pass filter eliminates lower and higher frequency components (see Figure 2-24(b) and Figure 2-24(c)). The value for m_l proposed by Okada was m1= 3 samples (for a signal sampled at 250 samples/sec, being equivalent to 12ms). As this value of m1 eliminated almost the entire V beat because of its low frequency, it had to be adjusted to m_l=12 samples (corresponding to 33 ms, @360Hz sampling rate). The peaks were made positive by squaring the output of the band pass filter. See Figure 2-24(c). All the peaks were obtained around the position of the QRS. The higher frequency components are enhanced (Figure 2-24(d)), leaving fewer high amplitude peaks on the closest area around the R wave peaks. In this case, m_2=4 samples (corresponding to 11 ms, @360Hz sampling rate) approximate the value Okada proposed in his work (m_2=m=3 samples, for a signal sampled at 250 samples/sec). The symmetry of y1 on the $2m_s+1$ window around each sample is checked (Figure 2-24(e)). In this case, $m_s=m_2$. Most of the redundant peaks (Those very close to the "peaks of interest") are eliminated in y4 because they correspond to no inflexion points in the curve. Still all of the beats have peaks in y4 at the moment they occur. Finally the pulse signal is generated based on y4. The threshold above which a peak is accepted is obtained following the same criterion Okada presented in his work (1/32) th of the maximal amplitude in the segment of 2m+1 samples around each point. Again $m=m_s=m_2$. Figure 2-24(f) shows a zoomed window to the signal to illustrate more clearly where the pulse lays on two kinds of beats. Note that the maximal value of y4 is precisely the closest to where the peak of the R wave is (this is shown in Table 2-8). The first part of the QRS is not included in this window, probably because in Figure 2-24(d) the window was not large enough to include its frequency components.



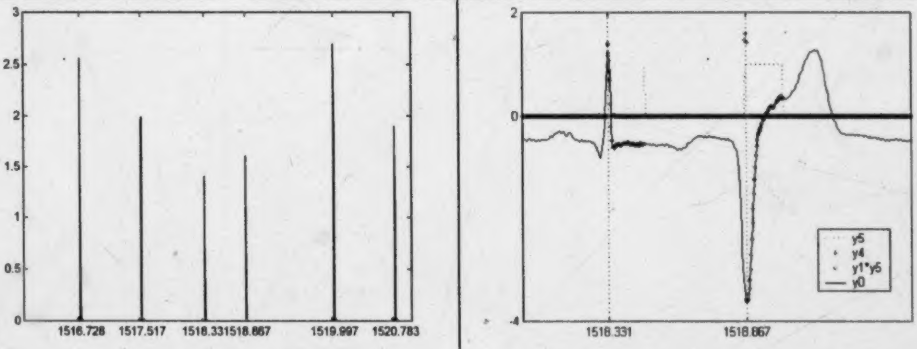
(a) The original signal y0 at 360 samples/sec. The marks in the horizontal axis correspond to the annotation of the peak of the R wave. There are 6 QRS class N, N, N, V, N, N in this order.

(b) The three point weighted average y1 (weights = [1 2 1] The (2m1+1) moving average y1b (m1=12)



(c) $y2 = (y1-y1b)^2$. It is a band pass filter, with its output squared.

(d) y3, where the higher frequencies are enhanced. $y3=y2*(2m2+1 \text{ moving average of } y2)^2$ (m2=3)



(e) y4, y3 points where y1 is symmetric in the 2m2+1 range.

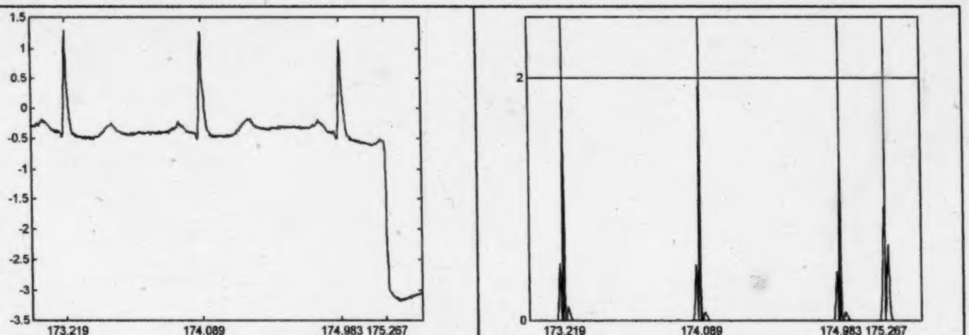
(f) y5 is 1 during 144ms starting from where y4 is greater than the threshold. Threshold=max(y4)/32. Note that the maximum value of y4 where y5 is 1 coincides with the annotation.

Figure 2-24: Okada's method: An illustrative example.

Table 2-8: Location of the R peaks by the annotator and by the Okada detector

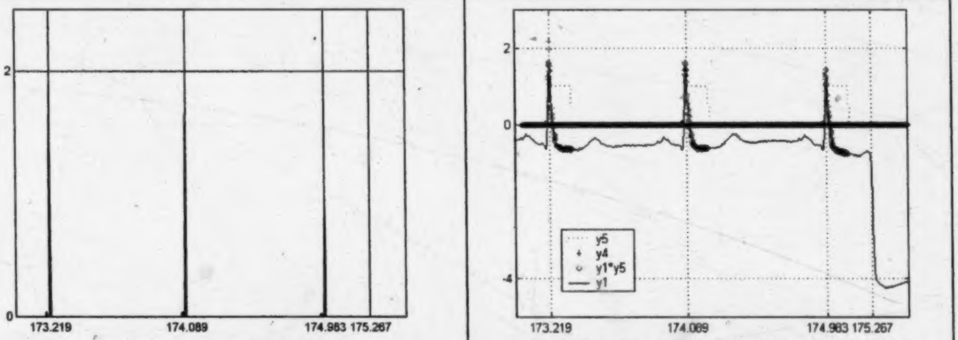
Position of peak (sec) As labeled.	Position of peak (sec) As detected.
1516.731	1516.731
1517.519	1517.519
1518.333	1518.333
1518.867	1518.869
1520.000	1520.000
1520.783	1520.783

In order to illustrate the case of a baseline drift, another extract of the MIT-BIH database was made, this time including a stepwise change in the baseline, as shown in Figure 2-25. The original signal Figure 2-25(a) shows three normal beats and a strong stepwise baseline wander. The first step is the band pass filtering. Details are not shown, but instead the final result is presented in Figure 2-25(b). This figure shows all the peaks made positive in y2. Note that there is a group of peaks at the position of the anomaly. These peaks are almost totally eliminated in Figure 2-25(c) when the symmetry in y1 (not shown but very similar to y0) is checked giving an almost "false-peaks"-clean signal y4. Finally, y5 is shown in Figure 2-25(d), which corresponds to the window that ignores the values of y4 below the threshold.



(a) Signal with three normal beats and a strong baseline wander (y0).

(b) Output of the band-pass filter squared (y2).



(c) Symmetric peaks only (y4).

(d) Detected QRS (y5) and signal (y1) where QRS is detected.

Figure 2-25: Example of Okada's method response in the case of a baseline drift.
a) The original signal y0. There are four normal beats. The baseline fluctuates abruptly after 2.2688 seconds. b) The squared output of the band pass filter y2. c) Enhancement of higher frequencies y3 and elimination of asymmetric elements obtaining y4. It is here where the effect of the baseline drift is eliminated. d) Elements above the threshold, from where the pulses are obtained. In this example, the sampling rate is 360Hz, m_1=12 samples, m_2=m_s=3 samples.

Table 2-9: Position of the detected and the labeled peaks using Okada's with a signal having a stepwise baseline drift

Position of peak (sec) As labeled.	Position of peak (sec) As detected.
173.219	173.219
174.089	174.086
174.983	174.981

To implement this algorithm, it is necessary to adjust several parameters, among which there are:

1. the characteristics of the digital band pass filter (basically, m_1).
2. the characteristics of the non-linear filter, that is the number of samples to be averaged and then squared ($2m_2$+1).
3. the estimated duration of a QRS in which the symmetry is checked ($2m_s$+1).
4. the threshold below which the signal y4 is forced to zero.

5. The length of the interval in which the non-zero points are grouped, corresponding to the estimate of the refractory period after a QRS is detected.

C. Pan - Tompkins and Hamilton - Tompkins algorithms for a real-time QRS detection.

This algorithm, described in detail in [1], [29], and [57], uses the slope, amplitude and width of the signal to detect QRS complexes in real time. As shown in Figure 2-26, this algorithm developed by Pan and Tompkins involves five steps. The signal (y0) is first passed through a band pass filter to maximize the QRS energy, estimated to be in the range of 5-15 Hz, while attenuating the rest of the components. These cut-off frequencies might vary depending on the case. After obtaining y1, the high slopes, characteristic to the QRS complexes are enhanced by the differentiation step (which output is y2). This differentiator should be linear up to a frequency safely greater than the high cut-off frequency of the band pass filter. In their work, the authors chose it to be 30Hz.

The third step consists in squaring y2 point by point (obtaining y3). In this way, the higher frequencies are enhanced and all the points are made positive before the next step.

Next, the signal goes through a moving window integrator. Here, the area below the waveform is calculated over an interval (m_l) set empirically by the authors to 150ms, the approximate duration of the widest QRS. The curve, therefore obtained, captures not only the steep slopes of normal QRS complexes but also large amplitudes and long durations characteristic of abnormal QRS complexes.

The final step is where the peaks of the R waves are detected. The objective is to find the window where to look for a peak in y1. To do so, the result of the moving average is analyzed. The signal is small all the time except where there is a QRS. In this case, it increases during the QRS; it maintains more or less its amplitude and then decreases again, forming a trapezoidal wave, as in Figure 2-27.

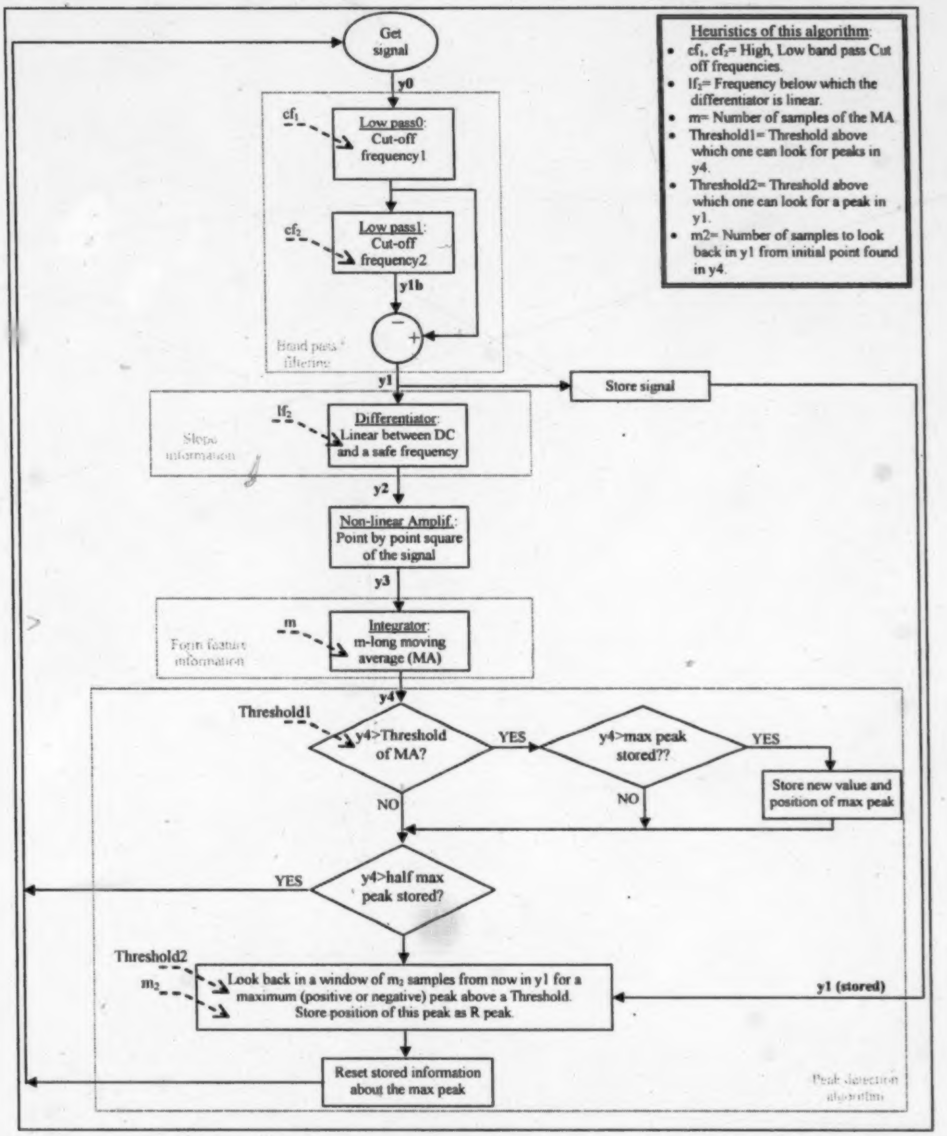

Get signal
y0
cf1
Low pass0: Cut-off frequency1
cf2
Low pass1: Cut-off frequency2
y1b
−
+
Band pass filtering
y1
Store signal
lf2
Differentiator: Linear between DC and a safe frequency
Slope information
y2
Non-linear Amplif.: Point by point square of the signal
y3
m
Integrator: m-long moving average (MA)
Form feature information
y4
Threshold1
y4>Threshold of MA?
YES
y4>max peak stored??
YES
Store new value and position of max peak
NO
NO
YES
y4>half max peak stored?
Threshold2
m2
Look back in a window of m2 samples from now in y1 for a maximum (positive or negative) peak above a Threshold. Store position of this peak as R peak.
y1 (stored)
Reset stored information about the max peak
Peak detection algorithm
Heuristics of this algorithm:
cf1, cf2= High, Low band pass Cut off frequencies.
lf2= Frequency below which the differentiator is linear.
m= Number of samples of the MA
Threshold1= Threshold above which one can look for peaks in y4.
Threshold2= Threshold above which one can look for a peak in y1.
m2= Number of samples to look back in y1 from initial point found in y4.

Figure 2-26: Flowchart of the Pan-Tompkins/Hamilton-Tompkins algorithm for QRS detection

An adaptive threshold (Threshold1 in Figure 2-26 and Figure 2-27) is used to search for the maximal value of y4. Based on this maximal value, the starting point of the backwards searching (bs1) is where the signal has decreased to a half of the maximum. Once this point has been found, the searching window will include all the samples from it back to a certain number of samples (m_2). The authors recommend a value of m_2 between 150ms and 250 ms.

After the window is found, y1 is analyzed in this segment to find a local maximum, which occurs at the peak of the R wave.

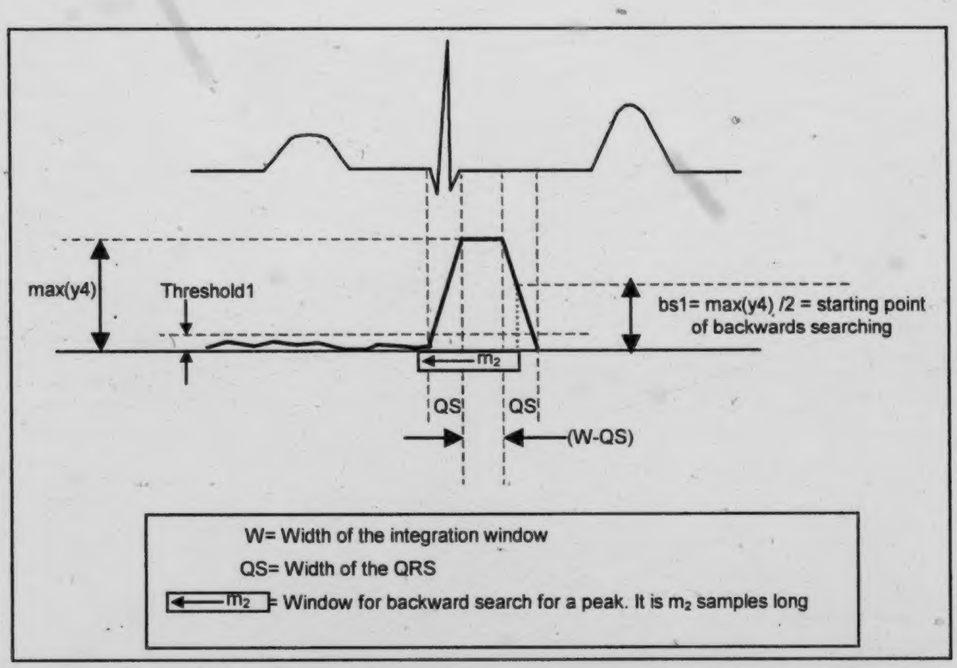

max(y4)
Threshold1
bs1= max(y4) /2 = starting point
of backwards searching
m_2
QS
QS
(W-QS)
W= Width of the integration window
QS= Width of the QRS
m_2 = Window for backward search for a peak. It is m_2 samples long

Figure 2-27: Expected relationship of a QRS complex to the moving integration waveform, exploited to detect the fiducial points

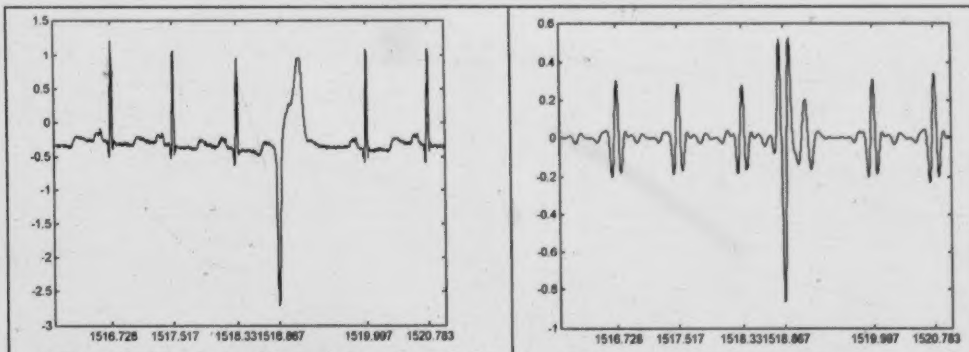
Top signal: typical shape of an ECG signal. Bottom signal: Simplified output of the moving window integrator. The positive slope occurs during the QRS. The peak of the signal is maintained for a time approximately equal to (W-QS), the width of the window minus the width of the QRS.

Each step of the procedure presented in Figure 2-26 is illustrated in a practical example shown in Figure 2-28. This example was extracted from the MIT-BIH database that will be described later on this work (Section 2.6.1). The original signal y0, sampled at 360Hz, contains six beats (Figure 2-28(a)): five normal beats (N) and one with a premature ventricular contraction (V). y0 goes through a band pass filter with cut-off frequencies of 5Hz and 11Hz (generating y1) to reduce the influence of any component outside this range of frequencies (see Figure 2-28(b)). Note that the biggest QRS, the premature ventricular beat, maintains its relative amplitude with respect to the normal beats, but its shape has changed: it is now symmetric with respect to its negative peak. The output of the previous step (y1) is then differentiated. With this differentiation, the new wave (y2) shown in Figure 2-28(c), allots more peaks (positives and negatives) than the previous one in the same space, while the space within groups of peaks contains less ripple. Even though the actual values of the signal are lesser, the proportion by which the higher peaks (groups of peaks around QRS location) relate to the smaller values (where there is no QRS nearby) is higher than in the previous step. When y2 is squared, it gives a no-negative signal (y3) with sharp positive peaks around the QRS locations (see Figure 2-28(d)). With the moving window integrator, the peaks accounted together to give a waveform feature. It can be observed from Figure 2-28(e) and Figure 2-28(d) that the rising slopes of the trapezoidal pulses coincide effectively with the occurrence of the QRS in y1. On each trapezoidal pulse, at the descending edge, the time of occurrence of nearest point to half the maximum value is selected as the end of the search window; the beginning of it is then 90 to 54 samples before it. In this implementation, 90 samples were used.

The last illustration (Figure 2-28(f)) shows the original signal, the pulse signal and the position of the detected peaks. These peaks are the maximum values of y1 over the windows determined earlier. To complete this information numerically and verify how far (or near) the detected peaks are to the marker, the position of both of them is presented below in Table 2-10. In this extract, the maximum difference between them is 3ms.

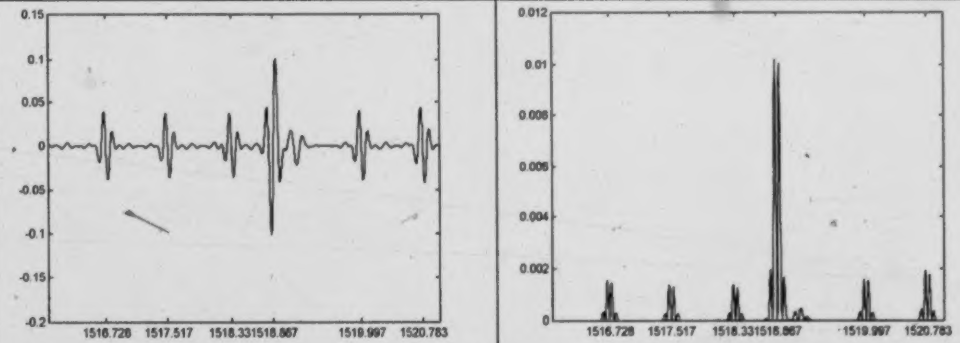
Table 2-10: Location of the R peaks by the annotator and by the Pan-Tompkins/Hamilton-Tompkins detector

Position of peak (sec) As labeled.	Position of peak (sec) As detected.
1516.728	1516.731
1517.517	1517.519
1518.330	1518.333
1518.867	1518.869
1519.997	1520.000
1520.783	1520.783



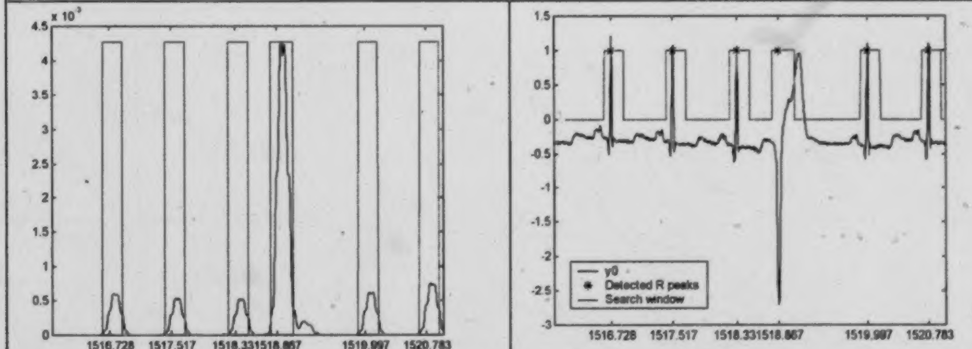
(a) The original signal y0 at 360 samples/sec. The marks in the horizontal axis correspond to the annotation of the peak of the R wave. There are 6 QRS belonging to class N, N, N, V, N, N, in this order.

(b) y1, the output of the band pass filter with cut-off frequencies 5Hz and 11Hz.



(c) y2, the output of the differentiator (Approximately linear up to 30Hz).

(d) y3, It consists simply in squaring y2.



(e) y4, the output of the integration step. A moving average was performed over a 54 samples window (~150ms). The pulses are obtained from detecting half of the descending slope (or the next inferior value).

(f) y5 is represented by the stars. It represents the location of the peaks of the R waves in each QRS. y0 and the pulses are shown for clarity.
Note the similarity between where the peaks were detected and where they were labeled to be. (Refer to Table 2-10)

Figure 2-28: Pan-Tompkins/Hamilton-Tompkins algorithm: an example

The example of a record with a baseline stepwise drift in Figure 2-29 is presented to illustrate the performance of the simplest version of the detector when faced with this situation. It can be observed that the step wandering is detected as a peak. This is why it would be interesting to apply here the rule about the symmetry of the signal presented by Okada.

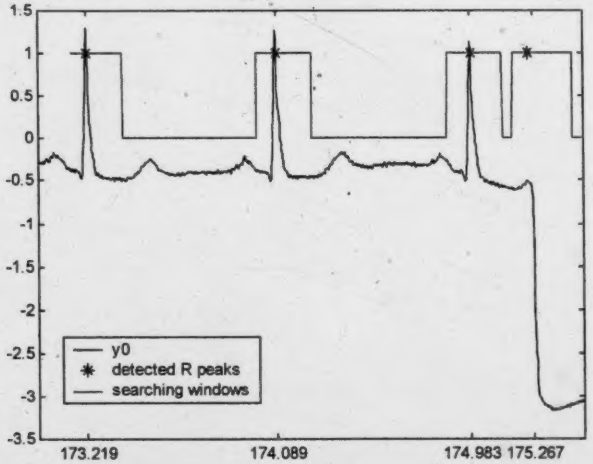


Figure 2-29: Example of Hamilton-Tompkins/Pan-Tompkins's method response in the case of a baseline drift

Table 2-11: Position of labeled and detected peaks in the case of a baseline drift using the Hamilton-Tompkins/Pan-Tompkins's method.

Position of peak (sec) As labeled.	Position of peak (sec) As detected.
173.219	173.222
174.089	174.092
174.983 (detected)	174.986 (Drift)
175.267	175.244

When this algorithm was tested [57] on the complete 24 Hour annotated MIT-BIH database, it gave a 0.68 % error. The error was calculated as

$$\frac{(FP + FN)}{TotalBeats} \tag{2-46}$$

where

FP (false positives) is the number of beats detected by mistake, FN (false negatives) is the number of beat that were missed, and TotalBeats the number total of beats in this study.

Note that to implement the algorithm described above it is required to adjust several parameters such as:

1. the characteristics of the band pass filter.
2. the length of the window in which the signal is to be integrated.
3. the length of the window where to search the peak.
4. the threshold on the moving averages where to look for the trapezoidal pulses.

D. MOBD algorithm

This algorithm, proposed by S. Suppappola and Y. Sun in [77] and [76], uses a non-linear transformation referred to as the Multiplication of Backward Differences (MOBD) to detect the QRS complex digitally.

x[n], the backward difference of a signal u at time n is the difference between the value of the signal at time n and its value at time (n-1):

$$x[n] = u[n] - u[n-1] \tag{2-47}$$

The N^{th} order MOBD non-linear transform at time n consists in the multiplication of the backward difference at time n with the previous (N-1) backward differences:

$$y[n] = \prod_{k=0}^{N-1} \left(x[n-k]\right) \tag{2-48}$$

A sign consistency constraint was added to the algorithm to force the output signal to zero if any two consecutive backwards differences in the N previous elements did not hold the same sign. In other words, for the sign consistency be satisfied at time n, all the previous N backward differences must have the same sign:

$$y[n] = 0 \text{, if } \operatorname{sgn}\left(x[n-k]\right) \neq \operatorname{sgn}\left(x[n-(k+1)]\right),\ k = 0,1,\ldots,N-2 \tag{2-49}$$

This optimization reduces the number of false peaks due to fluctuations in the signal, but it might also eliminate relevant peaks in very noisy signals causing it to fail to detect them as QRS.

The algorithm presented by Suppappola and Sun to detect QRS using the MOBD transform is sketched in Figure 2-30.

The signal is optionally passed through a quantizer to reduce its resolution.

Afterwards, the N^{th} order MOBD transformation with sign constraint is performed.

Data is only acquired while the signal is in the refractory period (RP). Starting at the end of the refractory period (EORP), the adaptive threshold is decremented by half each 100 milliseconds, up to the estimated value of the Average Noise (AN). If the processed signal is greater than the threshold, a peak is detected and the Refractory Period starts.

The next threshold is the maximum value of the MOBD transform from the moment the QRS was detected to the end of the refractory period.

The new estimate of the average noise is set to be the sum of half the previous average and the geometric mean of the signal from the previous EORP to the detection of the QRS.

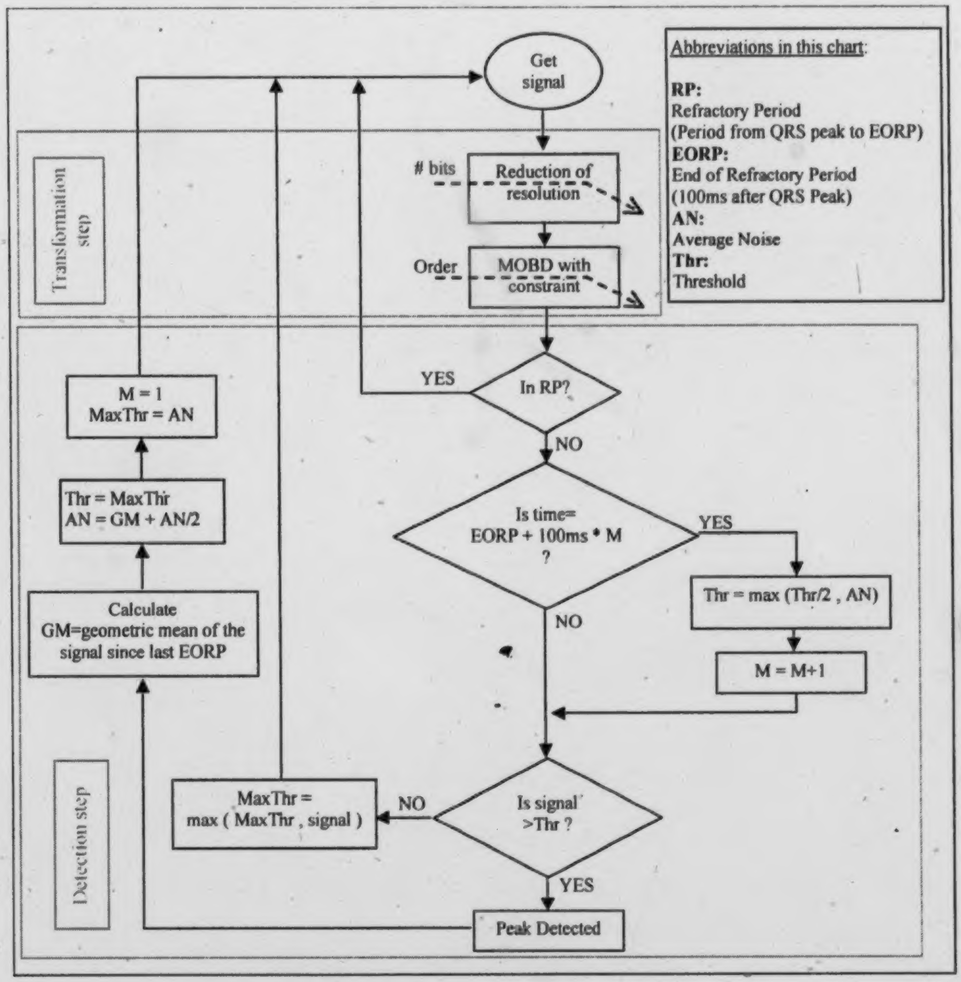

Abbreviations in this chart:
RP:
Refractory Period
(Period from QRS peak to EORP)
EORP:
End of Refractory Period
(100ms after QRS Peak)
AN:
Average Noise
Thr:
Threshold
Get signal
Transformation step
# bits
Reduction of resolution
Order
MOBD with constraint
In RP?
YES
NO
M = 1
MaxThr = AN
Thr = MaxThr
AN = GM + AN/2
Calculate GM=geometric mean of the signal since last EORP
Is time= EORP + 100ms * M ?
YES
NO
Thr = max (Thr/2 , AN)
M = M+1
Detection step
MaxThr = max (MaxThr , signal)
NO
Is signal >Thr ?
YES
Peak Detected

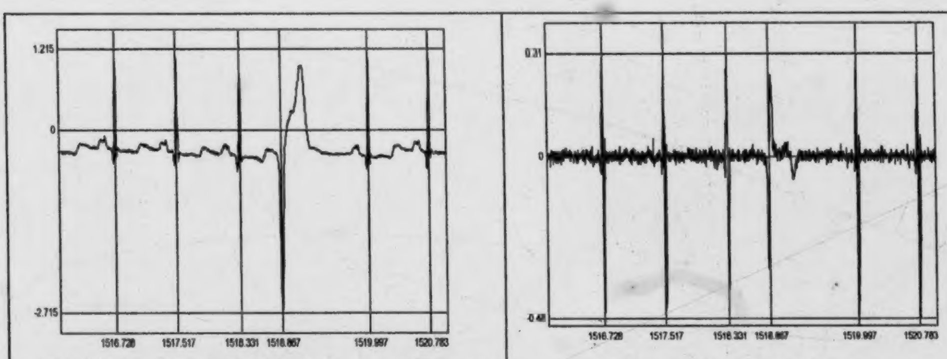
Figure 2-30: Flowchart of MOBD algorithm

There are several heuristic conditionals in this algorithm:

1. the quantization parameters; or in other words, how and how much the signal resolution is decreased.
2. the order of the MOBD transformation with or without sign constraint.
3. the threshold above which a peak is detected and how often this threshold is updated.

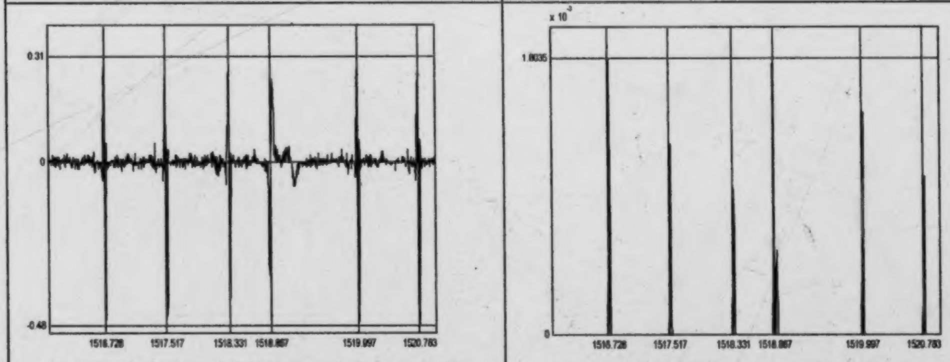
An example of the implementation of this method to detect QRS is presented in Figure 2-31. The first image, Figure 2-31(a), corresponds to the original signal, with five normal beats (on 1516.728, 1517.517, 1518.331, 1519.997, and 1520.783) and one premature ventricular beat (on 1518.867). Because the Nth order MOBD at each moment is the multiplication of the N previous first-order backward differences, then all of them were performed and showed on Figure 2-31(b). The sign constraint was then added on each one of the backward differences in Figure 2-31(c), setting to zero the backward differences whose sign is not the same to the previous one. Later, the Nth order transform (Figure 2-31(d)) will be the multiplication of the N previous elements of this wave, by applying equation **(2-49)** on equation **(2-48)**. Finally, the detection of the peaks is performed. There is an adaptive threshold above which a peak is detected. When this happens, a refractory period is set, and the new threshold is the maximum value of the transform during this period. After that, the threshold is decreased to half each 100ms, up to the estimated average noise. The average noise level is updated each time a peak is detected. It is calculated as the geometric mean of the transform from the previous end of refractory period to the detection of the peak. Note that at the end, the threshold that triggers the peak detection is not very different from beat to beat. But if there were some spurious peaks, they would not be detected as easily as if the threshold were constant.

In order to show a numeric reference of the performance of the detection, the position of the peaks (both of the labeled and of the detected) is presented in Table 2-12. This table shows that the R peaks are detected an average of 12 milliseconds in advance. In their work, Sun and Suppappola consider a valid detection interval, which begins 50 ms before and ends 100 ms after the annotation time mark. If this interval is taken as reference, then the detection is in an acceptable range.



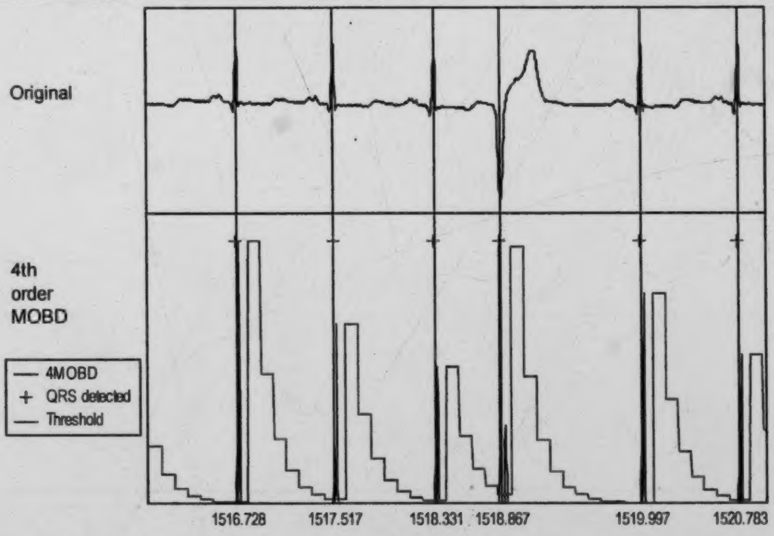
a) Original signal sampled at 360 samples per sec. There are six QRS: five normal (N) and one with a premature ventricular contraction (V). The vertical grid lines show the location of the R peaks.

b) First order backward difference.



c) First order backward difference with sign constraint.

d) Fourth order MOBD transformation with constraint.



e) Peak detection and threshold values based on the 4^{Th} order MOBD transform. The original signal is presented for reference.

Figure 2-31: Example of a 4^{th} order MOBD implementation

Table 2-12: Location of the R peaks by the annotator and by the MOBD detector in the first example.

Position of peak (sec) As labeled.	Position of QRS (sec) As detected.
1516.728	1516.717
1517.517	1517.508
1518.330	1518.322
1518.867	1518.847
1519.997	1519.986
1520.783	1520.769

When faced with a stepwise baseline drift this algorithm detects it as a QRS. This is illustrated in Figure 2-32 and Table 2-13. An additional rule should be included to the algorithm that complements the sign constraint. The sign constraint eliminates the oscillations due to changes in the direction of the wave in a very short period (three samples); but these changes are necessary in a longer period, because of the symmetry of the waves that form the QRS complex. A sign constraint should be applied, similar to the one Okada applies in his algorithm when he checks for symmetry with respect to the time axis in a period estimated to be the average duration of the waves that form the QRS.

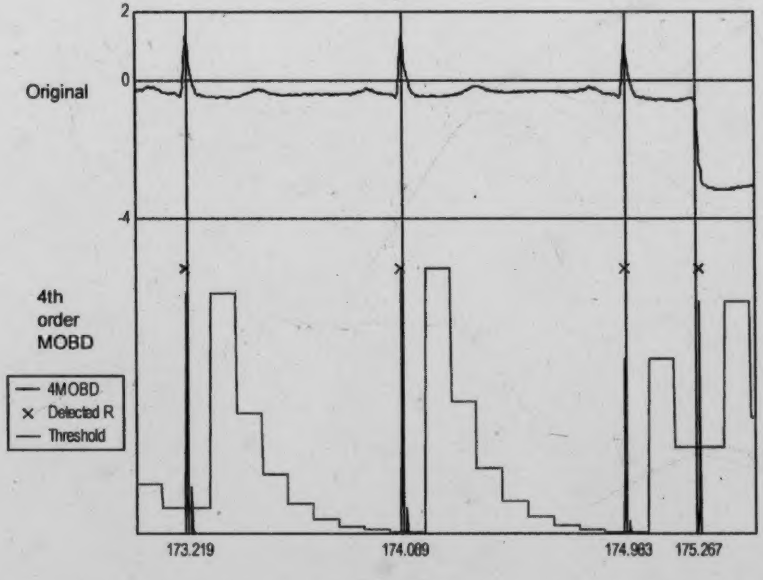


Figure 2-32: Example of MOBD QRS detector when there is a stepwise baseline drift.

Table 2-13: Position of labeled and detected peaks in the case of a baseline drift using the MOBD method

Position of peak (sec) As labeled.	Position of peak (sec) As detected.
173.219	173.211
174.089	174.078
174.983	174.975
175.267 (drift)	175.278

E. Detector using artificial neural networks

The algorithm described in this Section employs an artificial neural network model, specifically a two-layers perceptron ([42], [6]), to detect the QRS complex in the digital ECG signal. The structure presented here is a slightly modified version than the one presented in [21].

The artificial neural network is trained on the first ten beats of each record of the MIT-BIH database [53], and then tested on the rest.

The configuration of the two-layers perceptron is presented in Figure 2-34. In a nutshell, the inputs go through two main layers of neurons. The output of the first layer, also called the hidden layer, is the linear combination of the weighted variables of the input on which a non-linear function is applied. The second layer is the linear combination of the weighted values resulting from the previous layer, on which once again a non-linear function is applied. This non-linear function is also called the activation function of the neuron.

The activation function chosen in this work is the logistic sigmoid:

$$f(\text{net}) = \frac{1}{1 + e^{-\lambda\, \text{net}}} \qquad (2\text{-}50)$$

where net is the input to the activation function and λ is a constant that molds the curve as shown in Figure 2-33. The smaller the value of λ, the smoother the curve becomes around the origin.

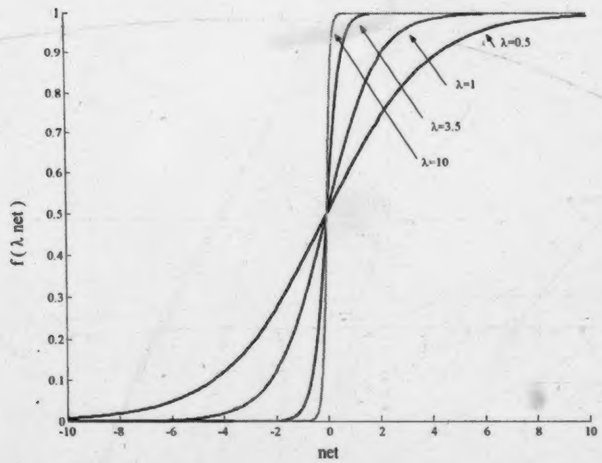


Figure 2-33: shape of the logistic sigmoid for different values of λ

A useful feature of this function is that its derivative can be expressed in a particularly simple form [6]:

$$f'(\mathrm{net}) = \lambda\, f(\mathrm{net})\left(1 - f(\mathrm{net})\right) \tag{2-51}$$

Its use will be explained later in this Section when the training process will be described.

The network goes as follows:

The R-dimensional input vector $\mathbf{x}=[x_1\ x_2\ \ldots\ x_R]$ is slightly changed to contain a constant term $x_0=1$ that will allow a bias to be added to the output.

Now the input vector is $\mathbf{x}=[1\ x_1\ x_2\ \ldots\ x_R]$, the output of the hidden layer is then:

$$y_s = f(\mathrm{netl}_s),\ \text{for s= 1, 2,..., S} \tag{2-52}$$

where

- S is the number of neurons of the hidden layer, and

$$\mathrm{netl}_s = \left(\sum_{r=0}^{R}(x_r\, w_{sr})\right),\ \text{for s= 1, 2,..., S} \tag{2-53}$$

where

w_{sr} is the weight assigned to x_r by neuron s.

Equations **(2-52)** and **(2-53)** are equivalent to:

$$\mathbf{y} = \mathbf{f}(\mathbf{W}\mathbf{x}) \qquad (2\text{-}54)$$

W is the S by (R+1) matrix of weights and **y** is the result of applying the non-linear function to each component of the vector obtained from the dot product between this matrix of weights and the input vector **x**.

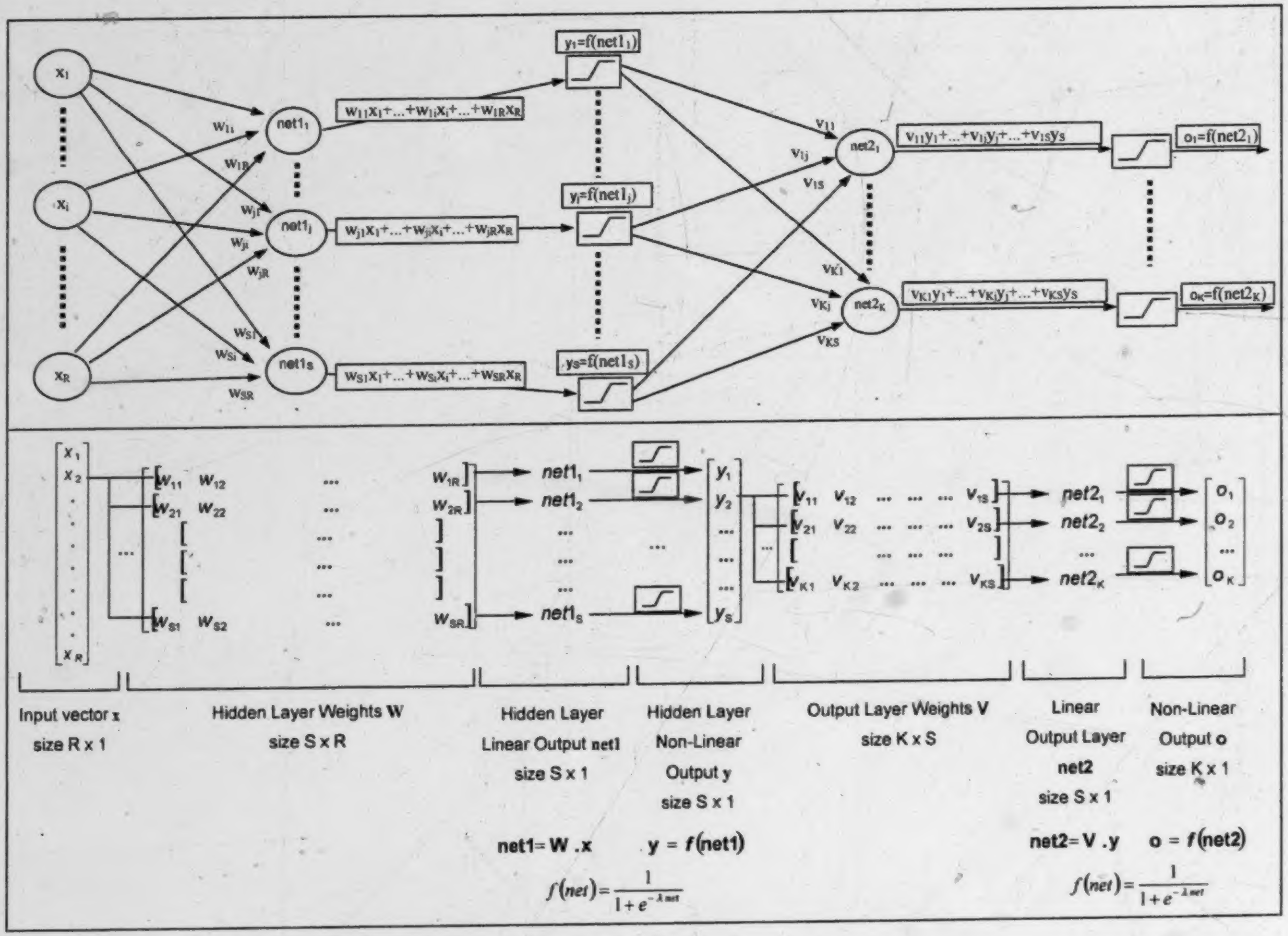

x1
xi
xR
w11
w1i
w1R
wj1
wji
wjR
wS1
wSi
wSR
net11
net1j
net1s
w11x1+...+w1ixi+...+w1RxR
wj1x1+...+wjixi+...+wjRxR
wS1x1+...+wSixi+...+wSRxR
y1=f(net11)
yj=f(net1j)
yS=f(net1S)
v11
v1j
v1S
vK1
vKj
vKS
net21
net2K
v11y1+...+v1jyj+...+v1SyS
vK1y1+...+vKjyj+...+vKSyS
o1=f(net21)
oK=f(net2K)
Input vector x
size R x 1
Hidden Layer Weights W
size S x R
Hidden Layer
Linear Output net1
size S x 1
Hidden Layer
Non-Linear
Output y
size S x 1
Output Layer Weights V
size K x S
Linear
Output Layer
net2
size S x 1
Non-Linear
Output o
size K x 1
net1= W .x
y = f(net1)
net2= V .y
o = f(net2)
f(net) = 1/(1+e^{-λnet})
f(net) = 1/(1+e^{-λnet})

Figure 2-34: Configuration of the two-layer perceptron

The outputs of the hidden layer are completed by a constant term ($y_0=1$) and become the inputs of the output layer. The result is expressed as follows:

$$o_k = f(net2_k), \text{ for } k = 1, 2, \ldots, K \tag{2-55}$$

where K is the number of neurons of the output layer and

$$net2_k = \left(\sum_{s=0}^{S} (y_s \, v_{ks}) \right), \text{ for } k = 1, 2, \ldots, K \tag{2-56}$$

V is the K by (S+1) matrix of weights and **o** is the result of applying the non-linear function to each component of the vector **net2** obtained from the dot product between this matrix of weights and the modified input vector **y**.

Training this network consists in finding the weight values that permit this structure to obtain outputs that fit the expected values.

The back-propagation method is used in this case to minimize an error function *Q* defined as half the sum of the squared differences between the desired outputs (do_k) and the actual outputs (o_k), namely:

$$Q = \frac{1}{2} \sum_{k=1}^{K} (do_k - o_k)^2 \tag{2-57}$$

The back-propagation is an iterative method by which the weights are updated according to the gradient of the error as follows:

$$\Delta v_{ks} = -\alpha \frac{\partial Q}{\partial v_{ks}} \tag{2-58}$$

$$\Delta w_{sr} = -\alpha \frac{\partial Q}{\partial w_{sr}} \tag{2-59}$$

where α is a positive learning rate. The value of this learning rate will affect the duration of the learning process: if α is greater, then the changes in the weights will be steeper but if it is smaller then the changes will be smoother. Both cases have pros and cons. The first option would be useful for reaching the minimum faster, but once there, it would make the output oscillate from iteration to iteration. The second option would be useful for fine-tuning the weights but it would slow down the learning process if the error function were far from the minimum.

To complete equation **(2-58)**, the gradient of the error with respect to the weights in the output layer has to be calculated.

From **(2-57)**, the first step in the deduction can be obtained:

$$\frac{\partial Q}{\partial v_{ks}} = \frac{\partial Q}{\partial O_k}\frac{\partial o_k}{\partial v_{ks}} = -\left(do_k - o_k\right)\frac{\partial o_k}{\partial v_{ks}}$$

Then from **(2-55)** and using the property in **(2-51)**, one gets:

$$\frac{\partial o_k}{\partial v_{ks}} = \frac{\partial o_k}{\partial net2_k}\frac{\partial net2_k}{\partial v_{ks}} = \lambda\, o_k\left(1 - o_k\right)\frac{\partial net2_k}{\partial v_{ks}}$$

Finally from **(2-56)**:

$$\frac{\partial net2_k}{\partial v_{ks}} = y_s$$

Subsequently, equation **(2-58)** can be expressed as:

$$\Delta v_{ks} = \alpha\left(do_k - o_k\right)\lambda\left(1 - o\right)o_k\, y_s \tag{2-60}$$

Now to complete equation **(2-59)**, the gradient of the error with respect to the weights in the hidden layer has to be calculated.

Once again, from **(2-57)**, the first step in this demonstration is obtained:

$$\frac{\partial Q}{\partial w_{sr}} = \frac{\partial Q}{\partial o_k}\frac{\partial o_k}{\partial w_{sr}} = -\sum_{k=1}^{K}\left(do_k - o_k\right)\frac{\partial o_k}{\partial w_{sr}}$$

Then from **(2-55)** and using **(2-51)**, one gets:

$$\frac{\partial o_k}{\partial w_{sr}} = \frac{\partial o_k}{\partial net2_k}\frac{\partial net2_k}{\partial w_{sr}} = \lambda\, o_k\left(1 - o_k\right)\frac{\partial net2_k}{\partial w_{sr}}$$

Once more, from **(2-56)**:

$$\frac{\partial net2_k}{\partial w_{sr}} = \frac{\partial net2_k}{\partial y_s}\frac{\partial y_s}{\partial w_{sr}} = v_{ks}\frac{\partial y_s}{\partial w_{sr}}$$

Now, from **(2-52)**:

$$\frac{\partial y_s}{\partial w_{sr}} = \lambda\left(1 - y_s\right)y_s\frac{\partial net1_s}{\partial w_{sr}}$$

And finally, from **(2-53)**:

$$\frac{\partial\, \mathrm{netl}_s}{\partial\, w_{sr}} = x_r$$

Subsequently, equation **(2-59)** can be expressed as:

$$\Delta w_{sr} = \lambda (1 - y_s) y_s\, x_r \sum_{k=1}^{K} (do_k - o_k) \lambda\, o_k (1 - o_k) v_{ks}$$

(2-61)

Training is an iterative process. One iteration corresponds to the input of one of the N R-dimensional vector of the training data set. When the N vector forming the training data set have been inputs of the network, then one epoch has occurred. The weights can be updated immediately right after each iteration (on line method) or after each epoch (off line method).

In the first case, equations (2-60) and (2-61) are used to update the weights in the new iterations from the values they had in the previous one:

$$\mathbf{W}(\mathrm{iter} + 1) = \mathbf{W}(\mathrm{iter}) + \Delta\, \mathbf{W}$$

(2-62)

and

$$\mathbf{V}(\mathrm{iter} + 1) = \mathbf{V}(\mathrm{iter}) + \Delta \mathbf{V}$$

(2-63)

In the case of the off line method, the weights in the new epoch follow the same previous equations using a variation equal to the sum of the variations accumulated on each iteration. Figure 2-35 presents a scheme of a training process in which the weights are updated on line. It stops when the error has reached a minimum or the weights do not change anymore. As with the other algorithms, the inputs could be pre-processed before entering the network. The examples that will be presented in this Section do not have any pre-processing. The input vectors are sent directly to the network.

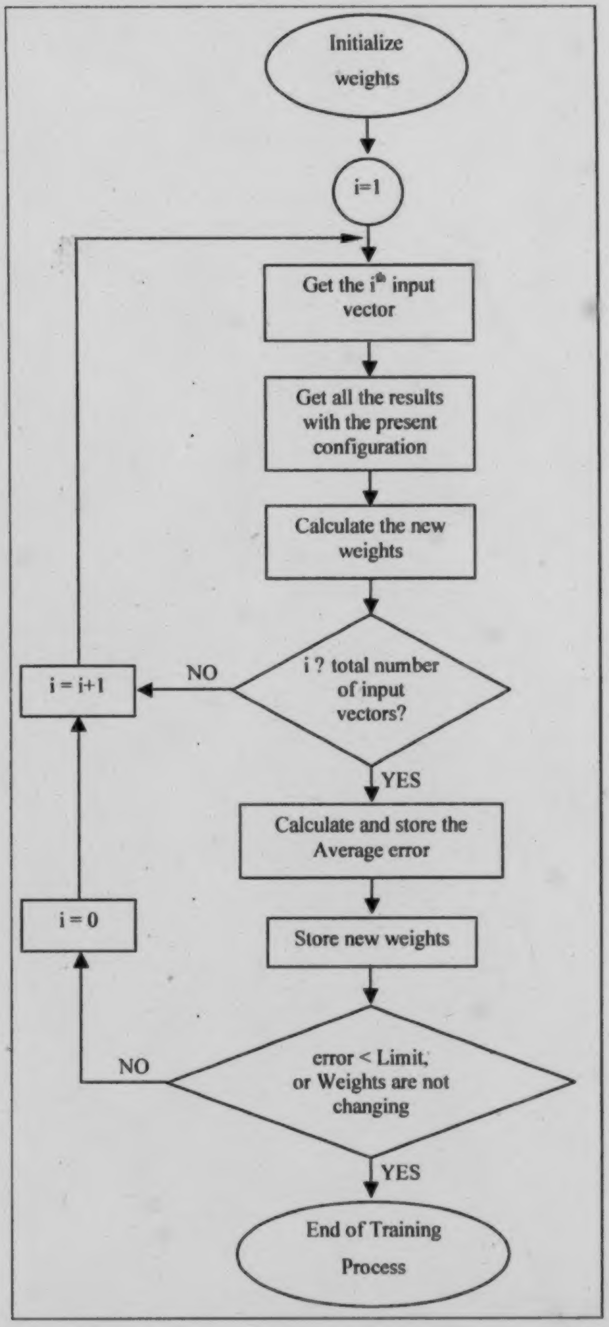

Initialize weights
i=1
Get the ith input vector
Get all the results with the present configuration
Calculate the new weights
i ? total number of input vectors?
NO
i = i+1
YES
Calculate and store the Average error
Store new weights
i = 0
error < Limit, or Weights are not changing
NO
YES
End of Training Process

Figure 2-35: Neural Network training process

At this point only the algorithm to create the model was explained. Implementing it for the QRS detection is described in the following example.

In this example, the training data set consists of a set of vectors that correspond to the values of the ECG in a 200 ms sliding window. The number of variables of the input vectors depends on the sampling rate of the signal. For example if the sampling rate was 360 Hz, then a 200 ms window corresponds to 73 samples.

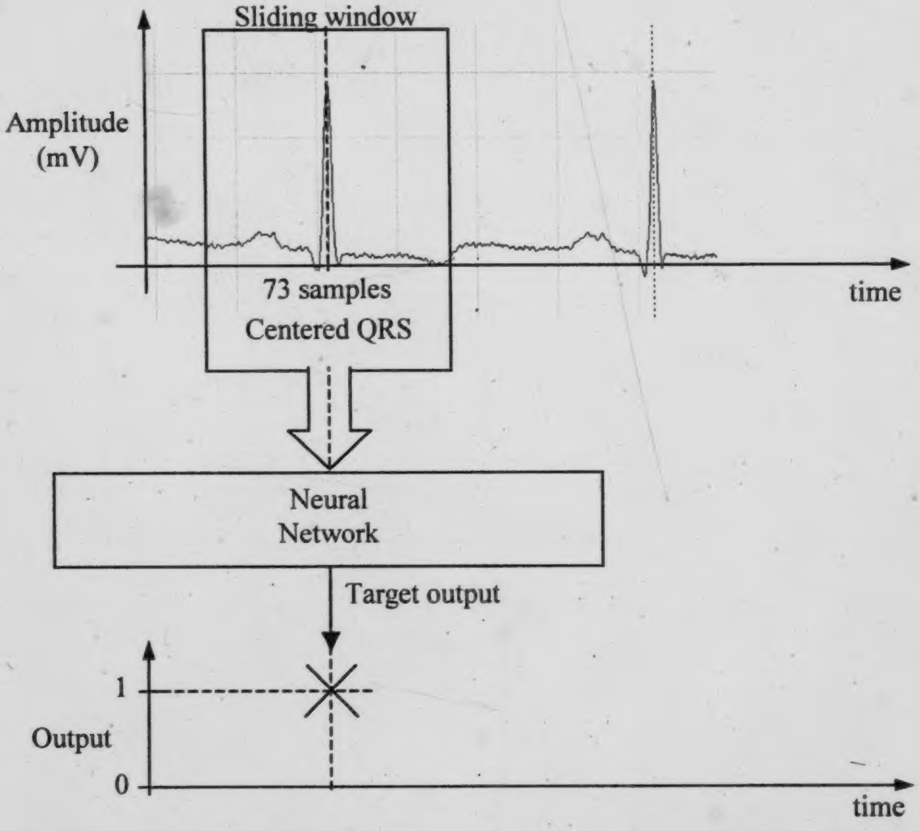


Figure 2-36: Inputs/Output of the neural network. Example for a centered QRS.

There is one output to the neural network. The desired output (or target output) for each one of these input vectors is either low when the QRS is not centered in the window or high around the moment the QRS is centered in the window. Because of the non-linear component in the output layer, the desired output is in the range [0,1].

There are numerous possibilities for setting the desired values in the outputs: One possibility is that the target output be 1 when the R peak is in the center of the input vector, and 0 otherwise. Another possibility is that the outputs could also be set in a more continuous way. Figure 2-37 shows one example of how the desired outputs can be set to train the network. In this example, the outputs get high gradually (proportional to the square of the distance to the borders of the window) in the range of plus and minus 100 milliseconds around the position of the peak of the R wave (which is considered in this work to be the center of the QRS).

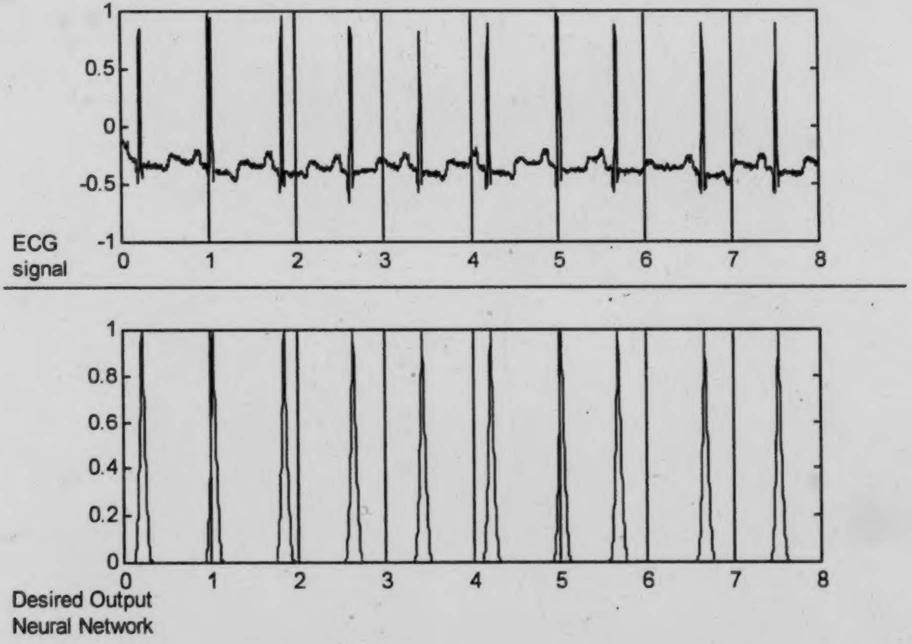


Figure 2-37: Desired Outputs for one example of training set

The outputs shown in this figure are the desired outputs for vectors coming from windows centered at the corresponding time frame.

Both type of outputs are explored in the experimental work (Section 4.3.3).

The training process is completed as in Figure 2-35 and the final result is a neural network with such weights that the model adequately assigns higher output values to the inputs containing centered QRS within them. This result of this particular example is illustrated in Figure 2-38.

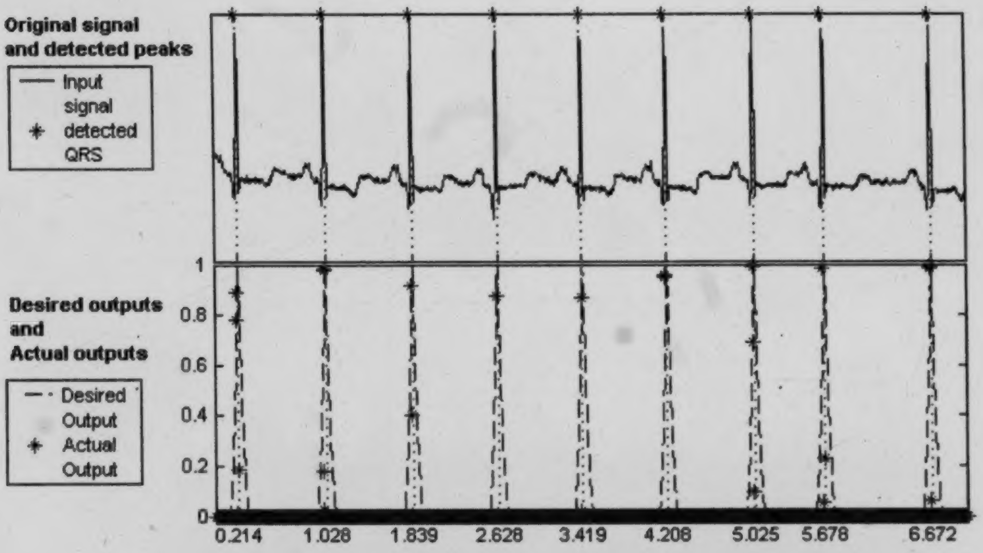


Figure 2-38: Result of the neural network on the training set of the example
The inputs from the original signal (top) originate outputs (stars in the bottom) that follow the tendency of the desired outputs (dashed line in the bottom figure): higher values around the peak of the R wave and lower values elsewhere.

Not only should the model be accurate within the training set, but it must also have the capability of generalization. In other words, if presented to a different group of data, it should be able to give the same kind of results: higher values around the QRS and lower values elsewhere.

To illustrate this, the model obtained with the training set of the example was tested on the rest of the record, and more particularly on the same segment the other algorithms were tried (see Figure 2-39). Note that this is just to illustrate very simply how the algorithm works.

From Figure 2-39, several observations can be made about the performance of the neural networks model. The first one is that with only normal beats in the training set, the model was able to generalize and recognize the PVC beat. But this same ability of generalize makes the model accept the steep T wave as a strong candidate to become a detected peak. Of course, the training set was in fact very reduced and no particular optimization was done to obtain this result, so that these features could be observed. Table 2-14 shows the location where the peaks were detected vs. the location they were supposed to be. All the distances are equal to 14 ms except for the PVC, which was detected 53 ms in advance with respect to the marker. It would be advisable to include a wider variety of QRS in the training set for the algorithm to perform more accurately.

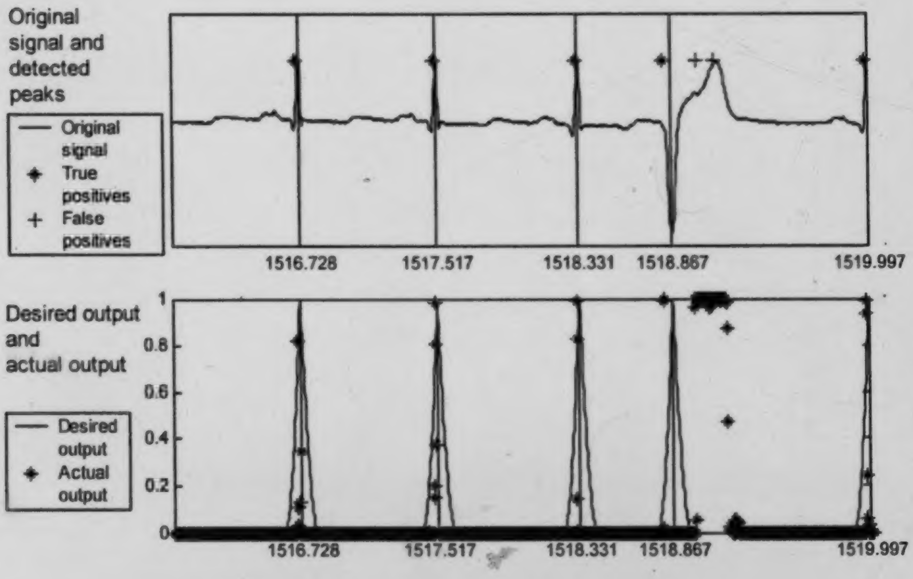


Figure 2-39: Example Neural Networks Model tested on new data
The original signal (continuous line on top figure) is shown with the detected peaks. All five peaks were detected but two peaks were detected that should not have been. Note that the actual output of the model shows an elevated density of higher values around the T wave that follows the premature ventricular beat.

Table 2-14: Location of the R peaks by the annotator and by the neural networks detector in the first example.

Position of peak (sec) As labeled.	Position of QRS (sec) As detected.	Difference in the position (sec)
1516.728	1516.714	0.014
1517.517	1517.503	0.014
1518.331	1518.317	0.014
1518.867	1518.814	0.053
1519.997	1519.983	0.014
N/A	1519.014	
N/A	1519.117	

When the model is faced with the case of stepwise baseline drift, it responds as desired by ignoring it. This is shown in Figure 2-40, where the highest outputs are once again around the position of the R peaks and the lowest everywhere else, including at the moment the drift appears.

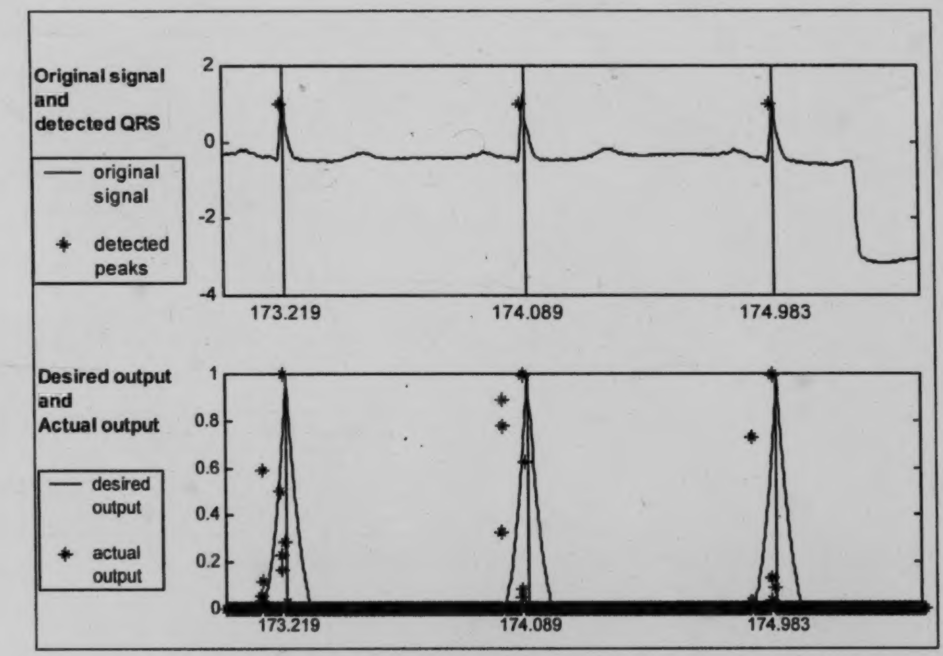


Figure 2-40: Example of neural network QRS detector when there is a stepwise baseline drift

The position of the detected and labeled peaks for this example is presented in Table 2-15. The peaks were detected on average 13 milliseconds earlier than the marker.

Table 2-15: Location of the R peaks by the annotator and by the neural networks detector in the case of a stepwise baseline drift.

Position of peak (sec) As labeled.	Position of QRS (sec) As detected.	Difference in the position
173.219	173.208	0.011
174.089	174.075	0.014
174.983	174.969	0.014

It seems that, with the implementation of the example, the peaks are always detected about a dozen milliseconds earlier than the time of the marker. If the exact position of the peak of the R wave is important, then this issue is worthy of exploration.

2.5.4. Classification of the QRS complex; related work

Electrocardiography has evolved since the time when the technique of ECG interpretation only contemplated the continuous visualization of the ECG tracing on the oscilloscope along with an automatic heart rate alarm [41].

It is evident that, to diagnose malfunctions of the human heart, ECG patterns must be correctly interpreted because it is in the shape of the ECG waveform and the heart rate that the state of cardiac health is generally reflected [63].

A lot of effort and methodological studies have been reported on the detection (see Section 2.5.3) and classification of ECG beats, which are the two main tasks of the temporal abstraction of the ECG waveform [8], along with the process to extract features that may make possible good results in the classification process.

In their search for ideal features from the digital ECG wave, diverse authors have proposed many options such as wavelet transforms, Hermite transforms, Discrete Fourier transforms, discrete cosine transforms, principal component analysis, trace segmentation, polygonal approximation, and others, which are sometimes combined with information about the heart rate.

The variety in the classification techniques proposed in the literature is also quite large; they go from knowledge-based techniques to a wide assortment of neural network configurations (Radial basis functions neural networks, multilayer perceptron, Intercepting sphere neural network, self organizing maps, fuzzy ARTMAP, and more).

A good reference for the implementation of several knowledge-based techniques is the review published in by Kundu et al. [41]. It is a comprehensive review of several knowledge-based techniques of interpretation of ECG signals: AND/OR graph, rule-based approach, and procedural semantic network. They also included certain syntactic approaches.

In this Section, some of other different algorithms found in the literature are presented with the type of features that are intended to discriminate the classes of QRS. They are presented in order, from the most recent to the earliest:

In their recent work, Carrault et al. [8] integrate numerical techniques for low-level perception (such as detecting P waves, and detecting and classifying QRS complexes), and symbolic techniques for high-level classification (Rhythm analysis). Their work consists of:

1) implementing the QRS detector proposed by Gritzadi in [27], (i.e. thresholding the result of a length transformation of the QRS signal).

2) designing and implementing a P-wave detector, based on wavelet decomposition in conjunction with an artificial neural network classifier.
3) extracting features of the detected QRS complexes using the coefficients of the wavelet transform of the signal.
4) classifying the beats into normal and abnormal, using a Radial basis functions neural network (RBF NN) on the wavelet transform coefficients.
5) analyzing the rhythm patterns in the ECG signal using inductive logic programming to discriminate between normal rhythms, left branch block, bigemy and Mobitz type II.

They used different extracts of the MIT-BIH database [53] (described in Section 2.6.1) for their different applications:

1) They had 7000 p-waves of records 100, 119, 214 and 231 annotated by cardiologists.
2) They used the records 100, 101, 121, 122, 214, 106, 119, 213, and 223, for a total of 16468 normal beats, 1657 premature, and 2002 left bundle branch block. They collected 1400 of these beats to train the PNN.

Carrault et al. [8] presented their classification results in terms of sensibility (99.3%) and specificity (98%).

Cuesta-Frau et al. [11] presented a work (similarly to [49] and [43]), whose motivation was to simplify the task of the experts who examine all the beats in Holter records (electrocardiograms of very long duration, commonly above 24 hours). They proposed the use of clustering techniques to reduce the number of beats to analyze. In this way only the prototypes of each cluster are presented to the user. Interestingly, in the clustering processes they describe, they compare segments of different lengths. Because of that, they do not implement the commonly used Minkowsky norms (see Table 3-2), which cannot be used to compare beats with different number of samples. Instead, they use a time-normalization or time alignment technique called the dynamic time warping (DTW) to obtain a dissimilarity measure. The work presented in [11] includes:

1) segmentation of the signal. They assume the position of the R wave is known. The beginning and end of each beat are placed at two given distances from the peak of the R wave, allowing some overlapping between consecutive beats.
2) feature extraction. They tested four types of features: raw information (the amplitudes obtained from the ECG signal), trace segmentation adapted as a

non-uniform sampling method for feature extraction, polygonal approximation, and wavelet transform.

3) pre-clustering. With this step, they intended to reduce the number of redundant beats to use in the clustering process. They calculated the dissimilarity between each beat and the previous N beats. If the dissimilarity between this beat and any of the previous N beats was below a conservative threshold, then this beat was considered redundant and it was discarded from the clustering process.
4) clustering. They tested two clustering algorithms: the Max-min and an adapted version of the k-means algorithm called the k-median (to cope with the use of dissimilarity measure instead of the Minkowsky norm)

They used 27 records of the MIT-BIH database for a total of 27412 beats that were reduced to 1026 beats after pre-clustering.

The results were evaluated in terms of error, defined as the number of beats that were mapped to clusters where their class was not the majority. They divided error into clustering error and critical error. The clustering error was computed as the percentage of beats misplaced but whose class was majority in some other clusters. Critical error was computed using the number of beats misplaced, whose class was not the majority in any cluster at all. Their work showed that the trace method for feature selection and the raw information gave the best results (around 10% overall error, with 9 clusters in both clustering algorithms). The best clustering algorithm was (not by far) the k-median.

Another work, by Rajendra Acharya et al. [63], compared the use of an artificial neural network and fuzzy equivalence relation for the classification of four cardiac disorders: Ischemic/dilated cardiomyopathy (class1), complete heart block (class2), sick sinus syndrome/atrial fibrillation/ectopics (class3), and normal (class4).

In their paper the authors explain how they made use of heart rate variability (HRV) as the base signal for analysis and classification of diseases. It is included in here to show other kinds of data from which to analyze the heart activity (other than ECG signal) and to mention an interesting technique to perform the classification. Their work consisted in:

1) feature extraction. They derived four parameters from the heart rate signal: Average heart rate in 10 min interval, ratio of the energy content in the band (33.3-100hz) and the energy content in the band (0-33hz), ratio of the energy content in the band (66.7-100hz) and the energy content in the band (0-66.7hz), the correlation

dimension factor obtained from the phase-space plot representing the heart rate in the horizontal axis and the delayed heart rate in the vertical axis.

2) classification using artificial neural networks [6].
3) classification using fuzzy equivalence relation. This process involves obtaining a fuzzy relation matrix for each class of data, and then comparing a fresh input with each group for classification [63].

The authors used a training set of 276 patient samples and a testing set of 66 patient samples. They presented their results in terms of correct classification per class:

1) using the artificial neural network: 85% for class1, 84% for class2, 90% for class3, and 85% for class4.
2) using the fuzzy equivalence classification: 90% for class1, 100% for class2, 95% for class3, and 95% for class4.

Dokur et al. [15] proposed a novel three-layer hybrid neural network for beat classification they called the intercepting spheres network (InS). Their work consisted in:

1) feature extraction and selection. They assumed the position of the R waves was known, and they extracted segments of 256 samples around the peak of the R waves. They used discrete Fourier transform and discrete wavelet transform on these segments, and selected eight coefficients to use in their application using divergence analysis.
2) comparison of classification results between their novel neural network, a multilayer perceptron (MLP) classifier and restricted coulomb energy (RCE) classifier.

   The InS network is a three layer neural network trained using genetic algorithms. The first layer is composed by neurons whose weights represent a single hypersphere in the input space of the network. The second layer is formed by nodes whose weights are codewords identifying classes, obtained in the training process (several codewords may identify the same class). The third layer combines the nodes of the previous layer into a single class node. The weights and number of nodes of the layers is determined in the training process using genetic algorithms.

Their work consisted in separate beats into ten different classes: N, L, R, V, P, p, a, F, f, and e (as defined in Table 2-19). 150 vectors of each class formed the training and testing sets. Their results are quite promising: 95.7% classification accuracy with the InS using the DWT features, and 78.2% classification accuracy with the InS using the DFT features.

The work of Maier et al. [49] in their "Unsupervised Morphological Classification of QRS complexes" reflected a similar purpose to the work presented in [11] and [43]: They did not attempt to build a classifier but rather grouped similar beats of the date set of study into few clusters for further analysis by the cardiologists. The following steps were taken to perform this study:

1) segmentation. They assumed the positions of the QRS segments were known. Around the peak of the R wave of each QRS complex, they took a window of 128 samples (corresponding to 355 ms of signal duration).
2) feature extraction. The experiments were carried out for three types of transformations of the data: Fast Fourier transform (4 coefficients for the real and the imaginary part), Discrete cosine transform (8 coefficients), and Hermite coefficients (7 coefficients and a time scaling parameter).
3) clustering. They used two different approaches: An agglomerative hierarchical cluster analysis, and a two-step correlation technique. The first one was applied on the transformed data, and the second one on the data with no transformation.

They intended to generate clusters that would group the data belonging to the same classes. They summarized the classes found in 48 records of the MIT-BIH (Table 2-18) database into four classes of clinical relevance recommended by the Association for the Advancement of Medical Instrumentation (AAMI).

To present their results, they computed the percentage of misclassified beats (the beats that did not belong to the class of the majority of the beats in their cluster), using either the categorization of the MIT-BIH database or its summarized version given by the AAMI.

They concluded the best features to use were the FFT coefficients: while these features did not make any significant difference in the classification performance, they allowed for less number of clusters and less residual classes (clusters that consisted of only one single element) than the other two kinds of features. Comparing the two different clustering approaches, the correlation gave the best classification results: for more than 90% of the data, the percentage of misclassifications was not higher than 0.5% (For the hierarchical clustering the smallest percentage of misclassification for the same percentage of the data was 2%).

Lagerholm et al. [43] presented their experiments of "Clustering ECG Complexes Using Hermite Functions and Self-Organizing Maps" as a way to group similar beats in clusters for further analysis. The purpose of this application is very similar to the ones presented in

[49] and [11]. Interestingly, and unlike any of the other works mentioned in this review, they used the data coming from both channels that the MIT-BIH database provides. Their work consisted of:

1) segmentation. They extracted the QRS segments using the envelope of the ECG [54]. The QRS segments were 200ms of duration with the peak of the R wave centered in the middle of the segment. Note that for each beat they had two sets of segments (one for each channel).
2) feature extraction. They described each segment by the first six coefficients of the Hermite transformation, along with the width factor associated to it. In addition, they included two features giving information about the distance among the R peaks to the previous and the next beat. In summary, each segment was described by ten features.
3) clustering. They used a 5 by 5 Self Organizing Maps configuration to perform the clustering of the whole data set.
4) comparison. Their results were compared with the work presented in [32].

All records of the MIT-BIH database formed the data set. Their results were presented as a gigantic confusion matrix for all the 16 classes of beats in the database. The misclassification percentage was also given: 1.5%, compared with the 94% accuracy given by the work of Hu et al. [32] (note that these two values are not directly related).

In their work, Hu et al. [32] presented a patient-adaptable ECG beat classifier using a mixture of experts approach. The motivation behind this work is the fact that "an ECG beat classifier which performs well for a given training database often fails when presented with a different patient's waveform" [32]. To alleviate this problem they proposed to divide the classification system into two combined classifier modules that are trained independently: One global expert (GE) to provide general classification ability and one local expert (LE) to provide the ability to recognize the patterns inherent on the patient. The basic notion is that linear combinations of several statistical estimates can perform better than any individual estimate. The expert network they proposed, called Mixture of Experts (MOE), was a combination of the GE and LE classifiers. Their work included:

1) feature extraction. They used the nine first principal components (selected by variance), the average time between R wave peaks over the previous 10 beats, the width of the QRS complex, and the encoded classification label of the beats.

2) classification into four classes resulting from the combination of annotations from the MIT-BIH database based on AAMI-recommended practice.

The experiments were performed on 33 records of the database (they excluded records with paced beats and records with no PVC). Records 100 to 124 were used to train the GE module, while records 200 to 234 were used to test the system. The LE module was trained with the 2.5 minutes of each record of the testing set.

The results are presented in terms of classification rate of each module and of the complete system: 62.2% for the GE only, 95.9% for the LE of each record only, and 94.0% for the MOE.

The study done by Ham and Han, in [28], proposed a system to classify cardiac arrhythmias using fuzzy adaptive resonance theory mapping (ARTMAP).

The data set consisted of an extract of the MIT-BIH database corresponding to records 116, 208, 210, 221, 228, and 233. They classified the beats into normal beats and premature ventricular contraction (PVC) beats. For this reason they only extracted the beats belonging to these classes. They assumed the position of the peak of the R wave was known. The segments to be classified consisted of 100 samples containing the peak of the R wave in the 51st sample. The features they used for such classification were two linear predictive coding (LPC) coefficients using Burg's method, and the mean square value of the segment (average signal power of the QRS segment). The results were given, among others, in terms of error rate (The ratio of the sum of misclassified or unidentified beats to the total number of beats tested): 0.671% in training and 0.83% in testing.

2.6. Description of the Data Set

In this thesis, some techniques are explored for detecting the QRS from the ECG signal, for modeling segments of this ECG signal, and for classifying the QRS segments of the ECG signal, modeled or in their raw form.

Different purposes called for different types of data sets. This is why three different extracts of the MIT-BIH database were used:

1. extract for QRS detection.
2. extract for Feature Selection.
3. extract for QRS classification.

The description of the data set includes first a general description of the MIT-BIH database, from where all the extracts were made. Then, the specifications of each one of the extracts is presented.

2.6.1. The MIT-BIH arrhythmia Database

The MIT-BIH arrhythmia database [50] [53] is a collection of 48 annotated records obtained by the Arrhythmia Laboratory of Beth Israel Hospital in Boston between 1975 and 1979. Two sets of records were both extracted from a set of more than 4000 Holter long-term recordings: The 100-series and the 200-series. The records of the first set were randomly chosen while the records of the second one were selected to integrate rare but important cases to the database. The identification numbers of all the records are shown in Table 2-16 for the 100-series and in Table 2-17 for the 200-series.

Each record of the MIT-BIH database is defined by three files that have the same name given by the record number but different extensions: One (*.dat) that contains the digitized ECG signal, one (*.atr) that contains the annotations and one (*.hea) that contains extra information about the patient.

The signal files contain two signals sampled at 360hz during slightly over 30 minutes; one coming from a modified limb lead II (MLII) and another coming mostly from a modified lead V1 (V2, V5 or V4 are seldom used instead of V1).

Table 2-16: Record numbers of the 100-series of the MIT-BIH arrhythmia database

100	105	111	116	122
101	106	112	117	123
102	107	113	118	124
103	108	114	119	
104	109	115	121	

Table 2-17: Record numbers of the 200-series of the MIT-BIH arrhythmia database

200	207	213	220	230
201	208	214	221	231
202	209	215	222	232
203	210	217	223	233
205	212	219	228	234

In the header files the leads used are specified, as well as the patient's age, sex, and medications. The annotation files contain the time when the R-wave peak of each beat occurs, the classification labels for each of these beats, some signal quality annotations, and a label when the rhythm changes.

There are 110588 beats in total; their distribution is shown on Table 2-18. The number of beats per record varies from 1518 to 3363.

Table 2-18: Beat types of the MIT-BIH arrhythmia database

Classification Label	Meaning	Number of beats with this label
N	Normal beat	75054
L	Left bundle branch block beat	8074
R	Right bundle branch block beat	7259
A	Atrial premature beat	2544
a	Aberrated atrial premature beat	150
J	Nodal (junctional) premature beat	83
S	Supraventricular premature beat	2
V	Premature ventricular contraction	7129
F	Fusion of ventricular and normal beat	803
!	Ventricular flutter wave	472
e	Atrial escape beat	16
j	Nodal (junctional) escape beat	229
E	Ventricular escape beat	106
P	Paced beat	7028
f	Fusion of paced and normal beat	982
p	Non-conducted P-wave	193
Q	Unclassifiable beat	464
	TOTAL	110588

In the MIT-BIH database, the information in the header files is stored in text format because of its short contents. On the other hand, the information in the annotation and signal files is stored using a special format to compress the data and to retrieve the data from these files a special library has to be used. To avoid this inconvenience, all the records were converted to text format by adapting two programs that were provided with the database (rdann and rdsamp).

The programs and libraries were included in the MIT-BIH Arrhythmia Database CD, third edition, May 1997. They could also be accessed from [24].

The compiled versions of the programs to convert the data to text could not be executed as given. The source codes were slightly adapted and compiled using a C++ Console Application in Metrowerks Codewarrior (version 3.2). To do so, a project was created that included all the default libraries for the C++ Console Application. Then the necessary source files (dbinit.c, calib.c, annot.c, signal.c, dbio.c, rdsamp.c and rdann.c), headers (ecgmap.h, ecgcodes.h, dblib.h, db.h) and libraries (wfdb32.lib) were copied to the same directory as was the project. The source files were included in the project in the same order they were specified above. The library "wfdb32.lib" was included in the corresponding Section of the project.

The changes in the source codes were simple. The simplest was to change the expressions of the non-standard "include" so that they would be read from the directory they were saved into. Also, the "io.h" header was added to the include Section of the file signal.c. Finally, the path for the input files and output files was changed so that the input files would be read from the same directory the executable was run from and the output files would be saved directly on the CD drive (D: /) instead of being only displayed on the screen. In order to generate the executable file to convert the annotation files (*.atr), all the source files but the rdsamp.c were included in the project. Similarly, the executable for converting the signal files (*.dat) to text was obtained by including all the source files but the rdann.c to the project.

The respective executables (one for converting the *.atr files and one for converting the *.dat files to text) were saved in the same directory as the database and then run from a MS DOS window. The text files were converted one by one and saved directly on CD. In this way two CDs were created: one for the 100-series and one for the 200-series. In these CDs, three files were saved for each record *RRR*:

- the header files (*RRR*.hea), already in text format.
- the text version of the annotation files (*RRR*a.txt), corresponding to the files (*RRR*.atr).
- the text version of the signal files (*RRR*.txt), corresponding to the files (*RRR*.dat).

The signal file consists in three columns containing approximately 650160 rows (corresponding to approximately 30 minutes recording sampled at 360Hz). The first column indicates the time in seconds in which each sample was taken. The second column indicates the voltage in mV of each sample that was recorded from the MLII lead. The third column contains the voltage in mV from the other lead, as mentioned in the previous Section.

The annotation file contains 7 columns corresponding to:

- the time of occurrence of the peak of the R waves, in minutes.
- the time of occurrence of the peak of the R waves, in seconds.
- the classification label of each beat, according to the symbols on Table 2-18

- three columns for signal quality
- rhythm label whenever the rhythm changes.

The size of each one of these files does not vary significantly from one record to the other. The size of the signal files in text format is of sixteen megabytes in average, while the annotation files occupy only between sixty and ninety kilobytes. When these files are converted to the text format, they approximately increase their size ten times. The header files maintain their size of one kilobyte.

The motivation for changing the format of the data is to be able to visualize easily this information without having to extract it from its compressed format each time.

It is from these text files that the actual data sets were extracted.

2.6.2. Data set for the study on QRS detection

In order to study the performance of the algorithms for detecting the QRS from the ECG signals, an extract of text files containing the annotations and the signal values were used, corresponding to the first ten minutes of each record.

The distribution of classes coincides with the description given below, in Table 2-19.

Table 2-19: Distribution of classes for the extract used to study QRS detection algorithms

Classification Label	Meaning	Number of beats with this label
N	Normal beat	24263
L	Left bundle branch block beat	2623
R	Right bundle branch block beat	2419
A	Atrial premature beat	698
a	Aberrated atrial premature beat	698
J	Nodal (junctional) premature beat	39
S	Supraventricular premature beat	0
V	Premature ventricular contraction	2168
F	Fusion of ventricular and normal beat	628
!	Ventricular flutter wave	127
e	Atrial escape beat	4
j	Nodal (junctional) escape beat	39
E	Ventricular escape beat	4
P	Paced beat	2158
f	Fusion of paced and normal beat	633
Q	Unclassifiable beat	182
	TOTAL	36683

The distribution of the classes per record is shown in Table 2-20, Table 2-21, Figure 2-41, and Figure 2-42:

Table 2-20: Distribution of classes per record; 100-series

Record Class	All	100	101	102	103	104	105	106	107	108	109	111	112	113	114	115	116	117	118	119	121	122	123	124
N	10266	750	623	95	690	100	787	566	0	537	0	0	827	559	502	615	733	489	0	503	590	811	489	0
L	1521	0	0	0	0	0	0	0	0	0	836	685	0	0	0	0	0	0	0	0	0	0	0	0
R	1163	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	708	0	0	0	0	455
A	36	5	1	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	27	0	0	0	0	0
a	36	5	1	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	27	0	0	0	0	0
J	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V	338	0	0	2	0	1	20	62	4	6	11	0	0	0	32	0	40	0	7	134	0	0	0	19
F	275	0	0	29	0	242	0	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0	1
!	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
j	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	1609	0	0	581	0	350	0	0	678	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f	280	0	0	29	0	245	0	0	0	1	2	0	0	0	2	0	0	0	0	0	0	0	0	1
Q	17	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
All	**15597**	760	625	736	690	955	807	628	682	544	850	685	827	565	538	615	773	489	769	637	590	811	489	532

Table 2-21: Distribution of classes per record; 200-series

Class\Record	All	200	201	202	203	205	207	208	209	210	212	213	214	215	217	219	220	221	222	223	228	230	231	232	233	234
N	13997	601	692	510	811	872	0	494	861	797	241	775	0	935	20	703	654	646	702	716	524	706	115	0	730	892
L	1102	0	0	0	0	0	451	0	0	0	0	0	651	0	0	0	0	0	0	0	0	0	0	0	0	0
R	1256	0	0	0	0	0	82	0	0	0	661	0	0	0	0	0	0	0	0	0	0	0	369	144	0	0
A	662	4	19	0	1	0	0	0	129	2	0	9	0	0	0	3	23	0	7	27	0	0	0	436	2	0
a	662	4	19	0	1	0	0	0	129	2	0	9	0	0	0	3	23	0	7	27	0	0	0	436	2	0
J	11	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V	1830	233	19	6	152	21	95	351	0	49	0	64	85	55	53	25	0	153	0	58	151	0	1	0	259	0
F	353	0	0	0	0	0	0	130	0	2	0	141	0	0	68	0	0	0	0	6	0	0	0	0	6	0
!	127	0	0	0	0	0	127	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0
j	11	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0
E	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0
P	549	0	0	0	0	0	0	0	0	0	0	0	0	0	549	0	0	0	0	0	0	0	0	0	0	0
f	353	0	0	0	0	0	0	130	0	2	0	141	0	0	68	0	0	0	0	6	0	0	0	0	6	0
Q	165	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	163	0	0	0
All	**21086**	842	761	516	965	893	755	1105	1119	854	902	1139	737	990	758	735	700	799	726	848	675	706	648	1016	1005	892

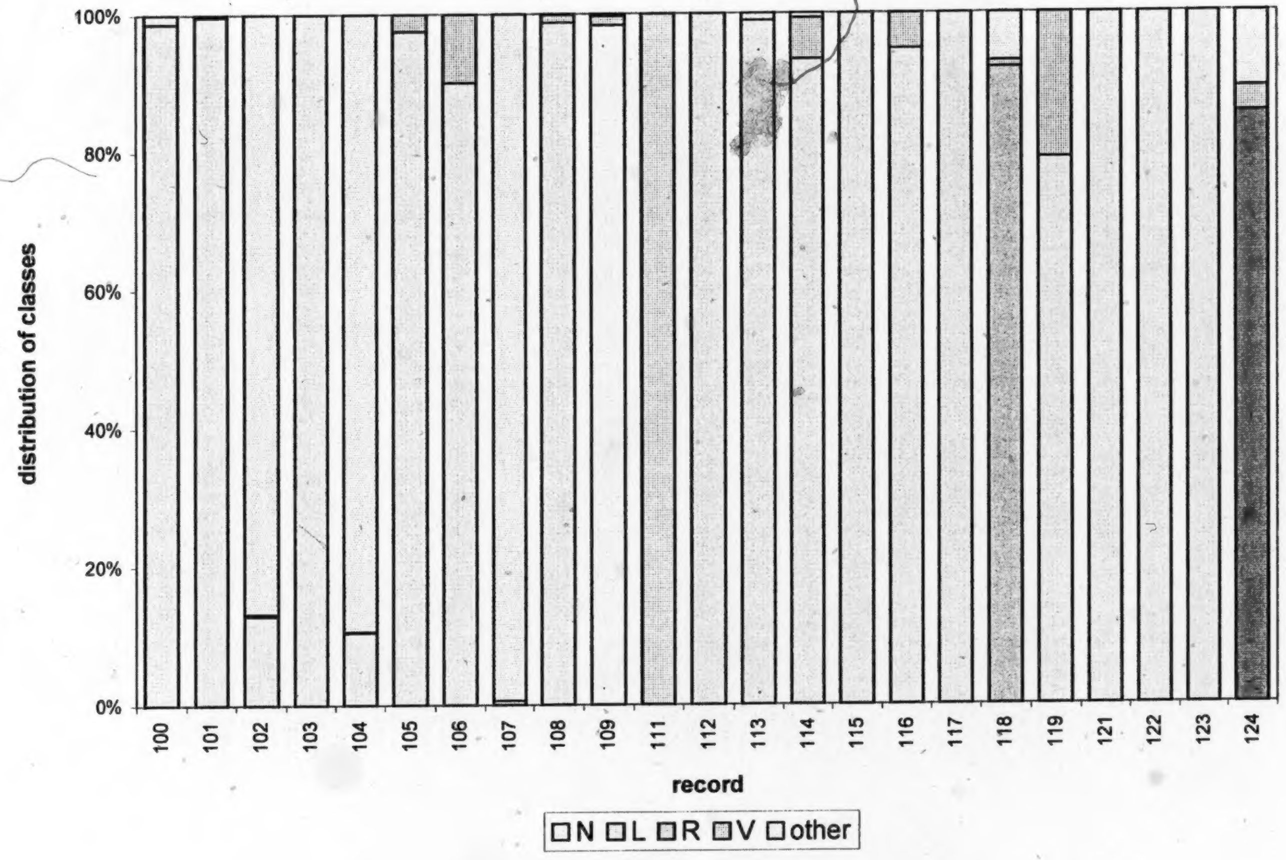


Figure 2-41: Distribution of classes per record; 100-series

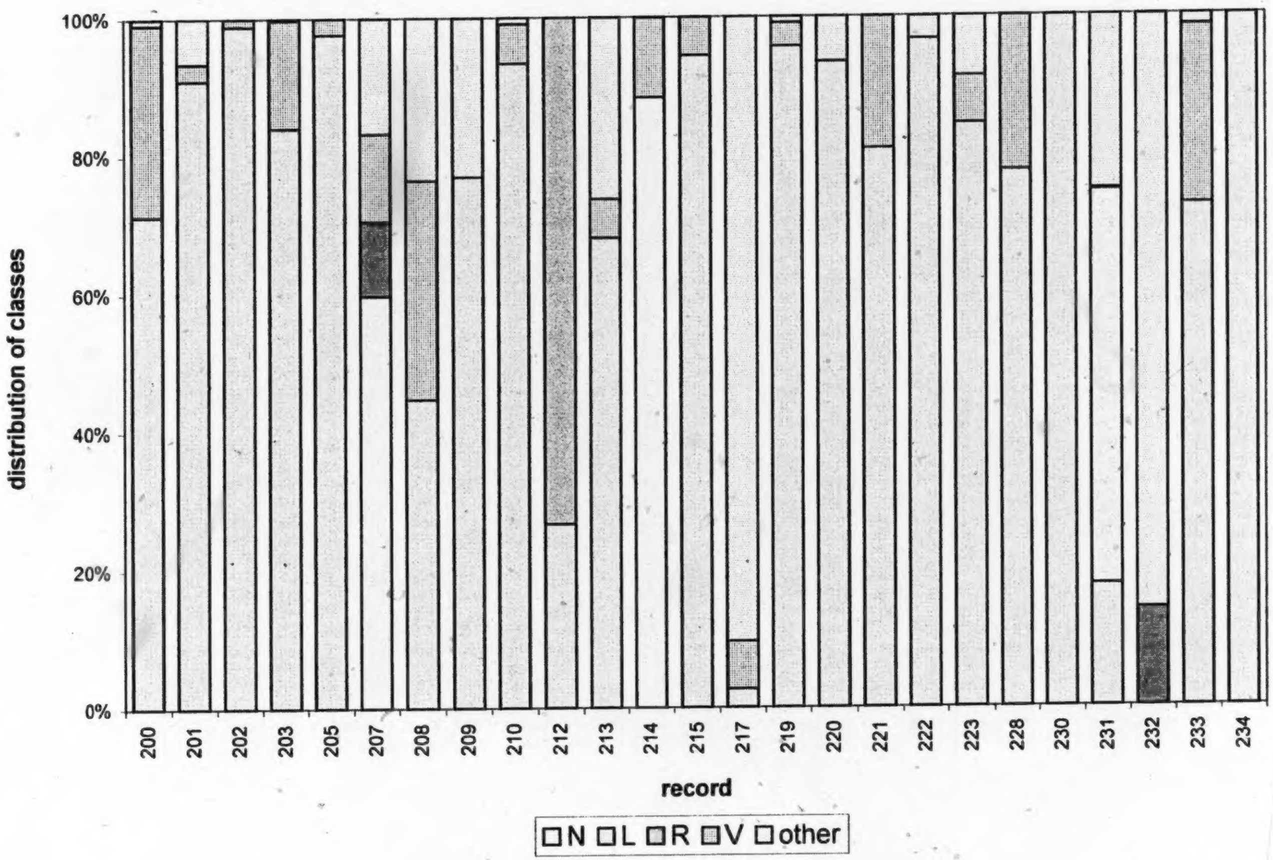


Figure 2-42: Distribution of classes per record; 200-series.

Within the different implementations of the algorithms, the target position of the peaks of R waves was extracted from the annotation files. And the signal information was processed as explained in Section 2.5.3., using the time of occurrence in seconds and the values on lead MLII.

2.6.3. Data set for the study on QRS classification

QRS classification implies analyzing the shape of the signal for a determined amount of time around the peak of the R wave. It also involves the knowledge of the time relationships with the previous and next R wave peak.

The first step in constructing the database for the study of QRS classification was to generate segments centered in the peak of the R wave of each QRS. To do that, both the signal files and the annotation files were used. The time of occurrence of the R peaks and the class of the QRS were obtained from the annotation files. The corresponding values in the MLII lead and the time of occurrence of each sample was obtained from the signal files.

All the beats present in the 48 records were extracted in the form of files containing two columns extracted from the signal file: the time of occurrence of each sample (in seconds) and the corresponding value on the MLII lead. The segments in these files are centered in the peak of the R-wave of the corresponding beat and completed with 36 samples before and after it, to form a segment of 73 samples corresponding to approximately 200 ms. The files were saved in a subdirectory corresponding to the record they were extracted from.

The name of these files was formed by the classification label of the beat (L), the number of the record (*RRR*) and the beat number in this record (*BBBB*), in the form L*RRR_BBBB*.txt. In this way, each file held the time information of its samples, the shape information in the form of voltage in the MLII lead and the label to classify the corresponding QRS.

After generating the 200ms-segments, it was necessary to complete the shape information with the R-R information, as the shape was not sufficient to discriminate between different classes of QRS (please, refer to Section 2.3.2).

The local rhythm information was included in the form of two measures: R^- and R^+. The values of these measures were calculated based on the definition presented in [43], namely:

$$R^- = R_i - R_{i-1} \qquad (2\text{-}64)$$

and

$$R^{+} = \Psi\left[\left(R^{-}_{i+1} - R^{-}_{i}\right) - \left(R^{-}_{i} - R^{-}_{i-1}\right)\right] \tag{2-65}$$

where

R_i is the time when the peak of the R wave of the ith beat occurs

$$\Psi(x) = \begin{cases} x, \text{if } x \geq 0 \\ 0, \text{if } x < 0 \end{cases}.$$

The measurements used in these equations can be visualized in Figure 2-43.

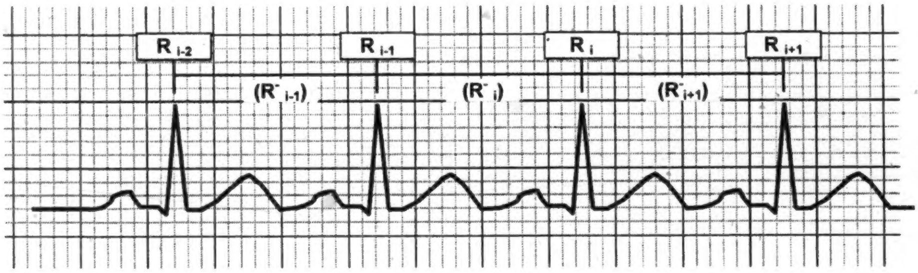


Figure 2-43: R-R measurements for obtaining R^- and R^+

R^- is the time since the previous beat.

R^+ shows how premature this beat is. It does not depend on the heart rate [43].

With the shape information and the local rhythm information, the data set for the QRS classification analysis was generated as follows.

The final data set was created as one file containing a table. A row of this table contained to the information of a determined beat:

- its identification in the format L*RRR_BBBB* ([Label][Record]_[Beat number]).
- seventy-three columns corresponding to the value on MLII of the 73 samples that form the 200ms-segments around the peaks of the R-waves.
- two columns corresponding to the R-R information held by R^- and R^+.

This data set contained an extract of the complete database corresponding to a total of 5310 beats:

- 1456 beats pertaining to class N.
- 835 beats pertaining to class V.
- 1456 beats pertaining to class R.
- 1593 beats pertaining to class L.

In summary, excluding their ID, vectors of 75 components represent the beats.

Note that during the experimental part of this work, this data set is used as it is or in its transformed version; but the beats represented there are the same in all cases.

2.6.4. Data set for features evaluation

In the case of the evaluation and extraction of features, these were performed to have reduced (and hopefully significant) features to represent the data to be classified. This data set was particularly used in Section 4.2.2 for determining the number of basis functions while approximating the segment by a Hermite series of expansion. For this study, because performing this transformation was very time consuming and machine demanding, a smaller data set was necessary. Because with this evaluation the number of features to approximate any segment was being determined, it was important to have an equal number of elements pertaining to the different classes.

This data set was formed by a total of 708 segments randomly extracted from the data set for QRS classification:

- 177 segments representing class N
- 177 segments representing class V
- 177 segments representing class R
- 177 segments representing class L

3. Classification Methods

3.1. Introduction

The classification problem can be approached in many ways depending on the structure the data has in its space. If the data set has a large number of features, the structure of the data is not always that obvious, and it is often useful to experiment by trial and error to select the best classifier for a specific group of data.

In this chapter, four classification techniques are described: the linear discriminant analysis (LDA), an adapted version of self organizing maps (SOM) to be used as a classifier, a radial basis function neural network (RBF NN), and a hybrid configuration in which clustering is performed by the SOM and the subsequent classification is performed by a RBF NN.

3.2. Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) can be described as a classifier strategy by which the data is projected onto a space that best separates it in the least squares sense using Multiple Discriminant Analysis (MDA). The data are then classified using the linear discriminant functions that are built for each class, based on the projection [18].

The goal of Multiple Discriminant Analysis (MDA) is to find a line or a group of lines onto which the data is projected to form well- separated (or at least better separated than in the original space) and compact clusters. These lines are obtained using the theory of Fisher's linear discriminant, generalized for the case of multiple classes.

Given **X**, a set of N d-dimensional vectors $\{\mathbf{x}_i\}$, for i=1,...,N; and given that, of these N samples, n_1 belong to class_1, n_2 to class_2 and so on up to class_c with n_c samples. Forming a linear combination of the d components of **x** yields the projection of its samples onto a line y in the direction of a transformation vector **w**, using the following equation:

$$\mathbf{y} = \mathbf{w}^T\mathbf{x} \tag{3-1}$$

If this idea were expanded to (c-1) linear combinations (assuming $d \geq c$), then the samples in **x** would be projected to (c-1) lines using (c-1) transformation vectors, as follows:

$$\mathbf{y}_i = \mathbf{w}_i^T\mathbf{x}, \quad i = 1,\ldots,c-1 \tag{3-2}$$

The previous equation can be written as a single matrix equation

$$\mathbf{Y} = \mathbf{W}^T\mathbf{x} \tag{3-3}$$

where

- **Y** is an N by (c-1) matrix that defines the data in the new (c-1)-dimensional space obtained applying equation (3-2).
- **W** is a d by (c-1) matrix which columns are vectors $\mathbf{w}_i$.

In this new (c-1)-dimensional space, the data set is still divided into subsets corresponding to the classes they belong to. The main idea in MDA is to find the direction of the lines in which these subsets are as well separated as possible.

This idea is illustrated in Figure 3-1 where two projections are shown for a simple case of c=2, d=2, N= 15 ($n_1 = 6$ and $n_2 = 9$). In the first projection, the samples belonging to both classes overlap widely (note that there is a segment in the line defined by **w** where the projection of five samples of each class fall onto). The second projection also includes some overlapping, but it is less severe than in the first case (only the projection of three samples belonging to class 1 overlap with the samples belonging to class 2). Following the criterion stated by the MDA, the transformation vector **w** of the second example would be chosen over the one in the first example. Note that this method is of little use if the distributions of the samples in the original space are multimodal and highly overlapping [18] because even an optimal transformation would not provide an adequate separation of the subsets in the new space.

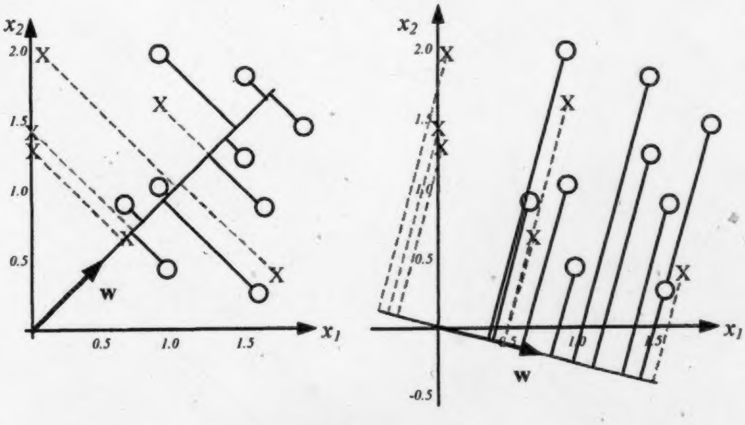


Figure 3-1: MDA-LDA example1: Projection of two-dimensional vectors onto two different lines [18]

The samples marked with an X belong to class 1; they are projected with dashed lines. The round markers represent samples that belong to class 2; they are projected with continuous lines.

To find the optimal transformation, two measures are introduced [17][51]: The within-class scatter matrix ($\mathbf{S}_W$) and the between-class scatter matrix ($\mathbf{S}_B$).

The within-class scatter matrix of **x** ($\mathbf{S}_W$) is proportional to the sample covariance for the pooled d-dimensional vectors that form the data set. $\mathbf{S}_W$ is given by:

$$\mathbf{S}_W = \sum_{j=1}^{c}\sum_{i=1}^{n_j}\left(\mathbf{x}_i^j - \boldsymbol{\mu}_j\right)\left(\mathbf{x}_i^j - \boldsymbol{\mu}_j\right)^T$$

(3-4)

where $\boldsymbol{\mu}_j$ is the mean of the n_j samples that belong to class j; and $\mathbf{x}_i^j$ is the ith sample pertaining to class j.

The between-class scatter matrix of **x** ($\mathbf{S}_B$) is defined as:

$$\mathbf{S}_B = \sum_{j=1}^{c} n_j(\boldsymbol{\mu}_j - \boldsymbol{\mu})(\boldsymbol{\mu}_j - \boldsymbol{\mu})^T$$

(3-5)

where $\boldsymbol{\mu}$ is the mean of the n samples that form **x**.

The set of samples in **x** projects to a corresponding set of samples in **y** using equations (3-2) and (3-3). The samples in the new space can be described by their mean vectors and scatter matrices. It has been demonstrated [17] that $\widetilde{\mathbf{S}}_W$ and $\widetilde{\mathbf{S}}_B$, the scatter matrices in the new space, can be expressed in terms of $\mathbf{S}_W$, $\mathbf{S}_B$ and the transformation matrix **W** as:

$$\widetilde{\mathbf{S}}_W = \mathbf{W}^T\mathbf{S}_W\mathbf{W}$$

(3-6)

and

$$\widetilde{\mathbf{S}}_B = \mathbf{W}^T\mathbf{S}_B\mathbf{W}$$

(3-7)

What is sought is a transformation matrix **W** that maximizes the between class scatter while minimizing the within class scatter. A way to do so is by maximizing the ratio of the between class scatter to the within class scatter. The determinants of the matrices $\widetilde{\mathbf{S}}_W$ and $\widetilde{\mathbf{S}}_B$ are taken as simple measures of between-class scatter and within class scatter respectively. The problem is then reduced to find the transformation matrix **W** that maximizes J, the ratio between the mentioned determinants defined as:

$$J(\mathbf{W}) = \frac{\left|\tilde{\mathbf{S}}_B\right|}{\left|\tilde{\mathbf{S}}_W\right|} = \frac{\left|\mathbf{W}^T \mathbf{S}_B \mathbf{W}\right|}{\left|\mathbf{W}^T \mathbf{S}_W \mathbf{W}\right|}$$

(3-8)

The columns of such matrix **W** are the generalized eigenvectors that correspond to the largest eigenvalues in:

$$\mathbf{S}_B \mathbf{w}_i = \lambda_i \mathbf{S}_W \mathbf{w}_i$$

(3-9)

If $\mathbf{S}_W$ is non singular, equation **(3-9)** becomes a conventional eigenvalue problem, which can be written as follows:

$$\mathbf{S}_W^{-1} \mathbf{S}_B \mathbf{W} = \boldsymbol{\lambda} \mathbf{W}$$

(3-10)

where $\boldsymbol{\lambda}$ and **W** are respectively the diagonal matrix of eigenvalues and the associated eigenvectors of $\mathbf{S}_W^{-1}\mathbf{S}_B$.

The mapping using MDA is illustrated by Figure 3-2. The data set in the example comprises samples represented by their values in the axes x1 and x2 (2-dimensional original space). The samples are divided into three classes (note that the number of classes exceeds the number of features). Ten samples belong to class1, eight to class2 and ten to class3.

The first step in the transformation is to find the transformation matrix **W** that allows the greatest separation between classes and the least separation of the samples within the classes. Using equations **(3-4)** and **(3-5)** the within class and between class scatter matrices. Then, equation **(3-10)** is employed to get the matrix of eigenvectors **W**. The columns of this matrix are the eigenvectors **w1** and **w2**, which define the direction of the lines y1 and y2 onto which the samples will be projected (Figure 3-2). Between the two eigenvectors **w1** was the eigenvector with the largest eigenvalue.

It is important to mention that in this case the maximum number of possible eigenvectors is limited by the number d and not by the number of classes. If there were 4 classes the number of eigenvectors would be also two.

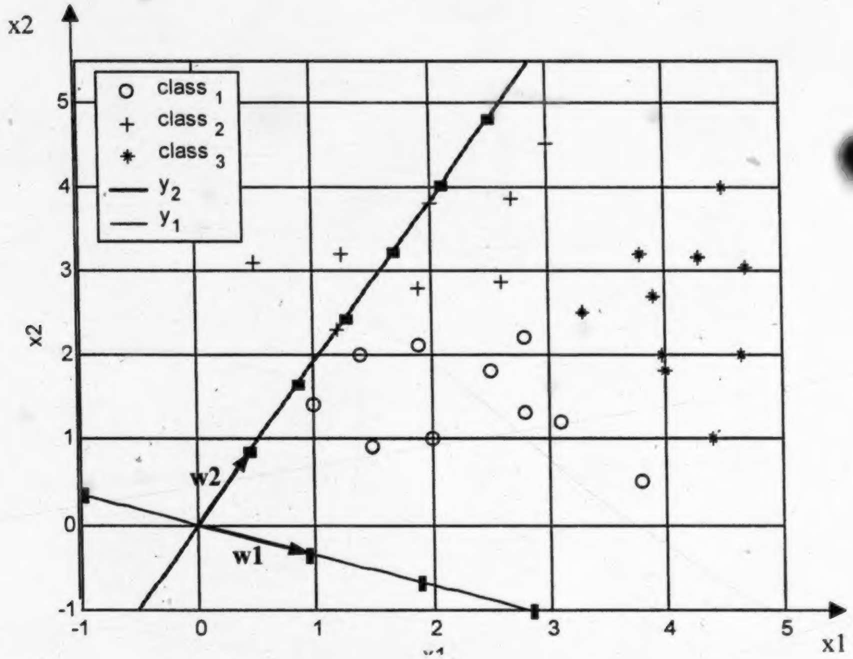


Figure 3-2: MDA-LDA example2: Definition in the original space of the projection lines y_1 and y_2 in the direction of w1 and w2

The approximated distribution per class on each axis (Assuming an unimodal normal density) is included in Figure 3-3 with the representation in the original space defined by the coordinates in x_1 and x_2.

The projection onto the new coordinates is shown in Figure 3-4. The approximated density distributions of the samples per class are shown for each new variable.

It is interesting to compare the distribution on the new axes to the distribution on the originals.

The distribution for x_2 (Figure 3-3) and the distribution for y_1 (Figure 3-4) are similar. In both cases there are two classes separated in some degree one from the other while one class overlaps them both.

The distribution for x_1 (Figure 3-3) and the distribution for y_2 (Figure 3-4) are also similar. In both cases one class (the one that overlapped both classes in the other axes) is separated from the other two, while the others highly overlap.

The main difference between the distributions in the original space and the distributions in the projections is that in the projections the overlap is reduced while the separation is increased.

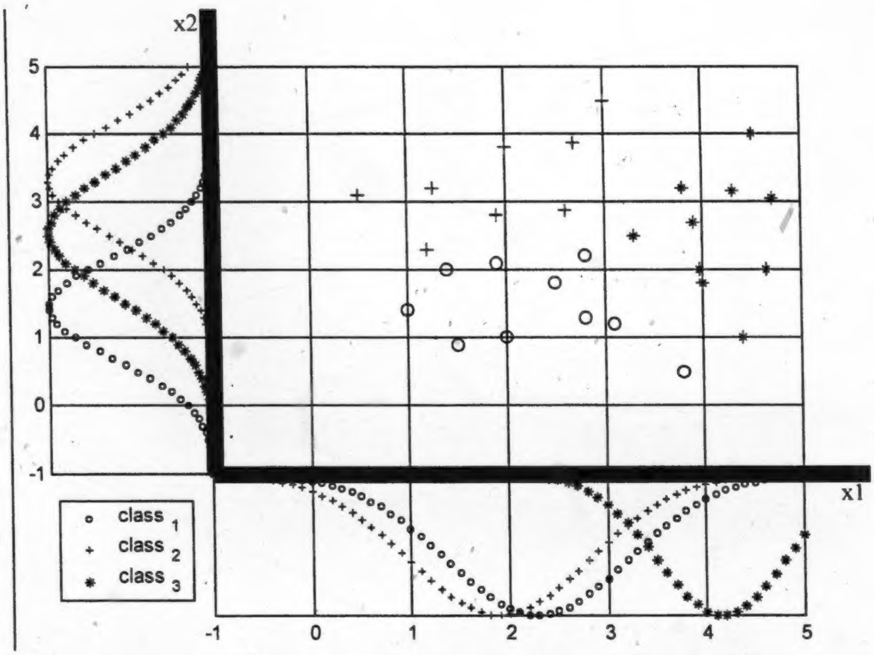


Figure 3-3: MDA-LDA example2: Data set of two-dimensional vectors belonging to three classes in the original space (each sample is defined by coordinates in x_1 and x_2)

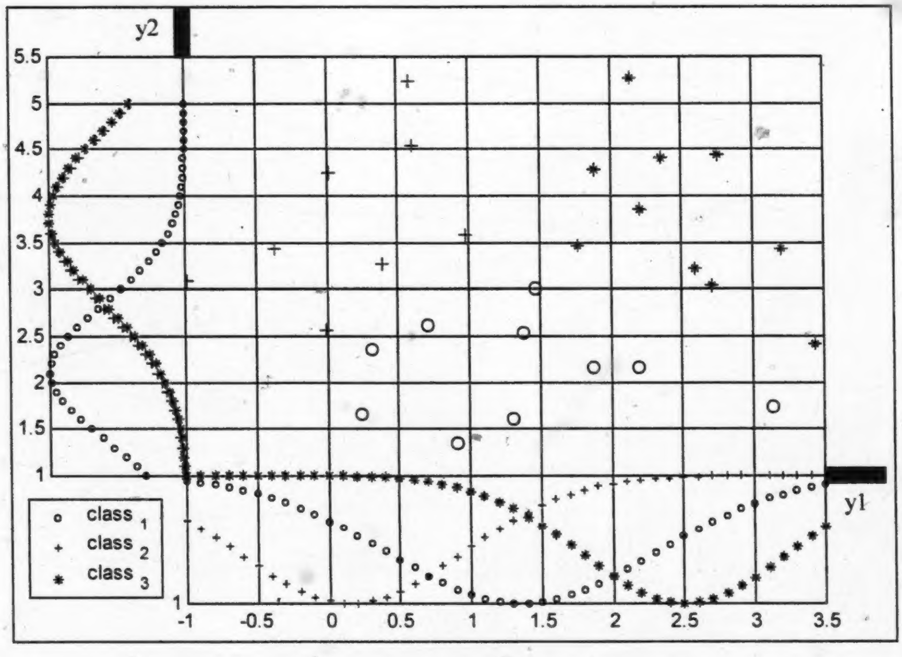


Figure 3-4: MDA-LDA example2: Data in the transformed space (each sample is defined now by coordinates in y_1 and y_2)

Once the distributions are projected to the optimal subspace (or space as in the MDA-LDA example2), then the full classifier is built.

The classification process is developed in terms of a set of discriminant functions [6][69][18] $g_i(\mathbf{x})$, $i = 1,\ldots,c$ such that a feature vector $\mathbf{x}$ is assigned to a class C_k if

$$g_k(\mathbf{x}) > g_j(\mathbf{x}) \qquad \text{for all } j \neq k \tag{3-11}$$

In other words, c discriminant functions are computed and the category corresponding to the largest discriminant is selected.

There is not a unique choice for the discriminant function. In this case, the decision rule to obtain a minimum error rate is cast by choosing:

$$g_i(\mathbf{x}) = P(C_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_i)\, P(C_i)}{\sum_{j=1}^{c} p(\mathbf{x} \mid C_j) P(C_j)} \tag{3-12}$$

where

- $P(C_i)$ represents the prior probability of class C_i
- $p(\mathbf{x} \mid C_i)$ represents the i th class conditional probability density function of $\mathbf{x}$.

Because the denominator in equation (3-12) does not depend on the classification label C_i, it does not affect the decision in the classification process. It can therefore be removed from the expression without changing the classification result. Equation (3-12) can be written in the form:

$$g_i(\mathbf{x}) = p(\mathbf{x} \mid C_i) P(C_i) \tag{3-13}$$

Generally, if every $g_i(\mathbf{x})$ is replaced by $f(g_i(\mathbf{x}))$, where $f(\cdot)$ is a monotonically increasing function the resulting classification does not change, since it is only the relative magnitudes of the discriminant functions that are important. If the function $f(\cdot)$ is chosen conveniently to be the natural logarithm, then the discriminant function becomes:

$$g_i(\mathbf{x}) = \ln[p(\mathbf{x} \mid C_i)] + \ln[P(C_i)] \tag{3-14}$$

The effect of the decision rule is to divide the feature space into c decision regions. If two regions are contiguous then the decision boundary separating them is located in the combinations of $\mathbf{x}$ where the discriminant functions of these regions have the same value.

One may assume the densities $p(\mathbf{x} \mid C_i)$ to be multivariate normal in the form:

$$N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right) \tag{3-15}$$

where

- $\mathbf{x}$ is a d-component column vector representing one sample.
- $\boldsymbol{\Sigma}$ is the d-by-d covariance matrix of the set of N samples.

The expression in equation (3-14) can therefore be evaluated promptly and be:

$$g_i(\mathbf{x}) = -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) - \frac{d}{2} \ln(2\pi) - \frac{1}{2} \ln |\boldsymbol{\Sigma}_i| + \ln(P(C_i)) \tag{3-16}$$

If it is assumed that all the classes share the same covariance, then the second and third terms of the equation above can be eliminated. Furthermore, if the prior probability of all the classes is the same, the only term that is maintained is proportional to the squared Mahalanobis distance from $\mathbf{x}$ to $\boldsymbol{\mu}_i$. The consequent decision rule assigns the feature vector $\mathbf{x}$ to the category of the nearest mean.

With some more manipulation, equation (3-16) can be expressed as a linear function[18][35]:

$$g_i(\mathbf{x}) = \mathbf{a}_i^T \mathbf{x} + b_{i0} \tag{3-17}$$

where

$$\mathbf{a}_i = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_i$$

$$b_{i0} = -\frac{1}{2} \boldsymbol{\mu}_i^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_i + P(C_i)$$

Finally, equation (3-17) is applied in the optimal space obtained earlier, which means that:

1. the covariance matrix in the optimal space is defined by the projections of $\mathbf{x}$ on $\mathbf{y}$.
2. the independent variable is not $\mathbf{x}$ but $\mathbf{y}$.
3. the mean of class i is also expressed in terms of its coordinates in the transformed space.

In the classifier application, the Linear Discriminant function implicitly contains the projection onto the optimal subspace. It may therefore be expressed directly in terms of the original coordinates as follows:

$$L_i(\mathbf{x}) = \boldsymbol{\mu}_i^T S_W^{-1}\left(\mathbf{x} - \frac{\boldsymbol{\mu}_i}{2}\right) + \ln\left(P\left(C_i\right)\right)$$

(3-18)

where

- S_W is defined as in equation (3-4),
- P(Ci) can be estimated as $\frac{1}{c}$ or as $\frac{n_i}{N}$ (the number of samples that belong to class i divided by the total number of samples)

Figure 3-5 shows the three discriminant functions evaluated for values of x_1 and x_2 in the range [-1 , 5]. Three regions can be clearly observed where each one of the functions has larger values than the others.

The data set and the boundaries separating each class within it are shown in Figure 3-6. In this example, all the samples in the region labeled class 1 will be classified into class 1, and so on.

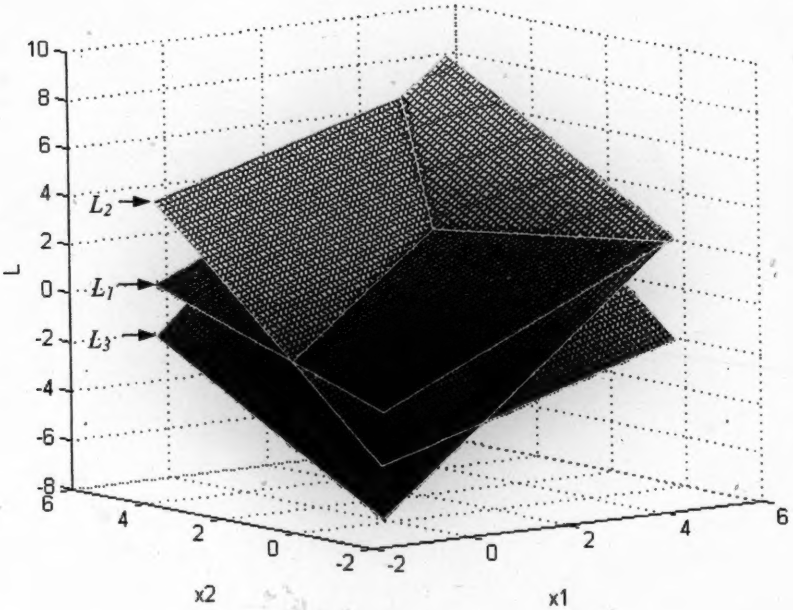


Figure 3-5: MDA-LDA example2: Discriminant functions for each class

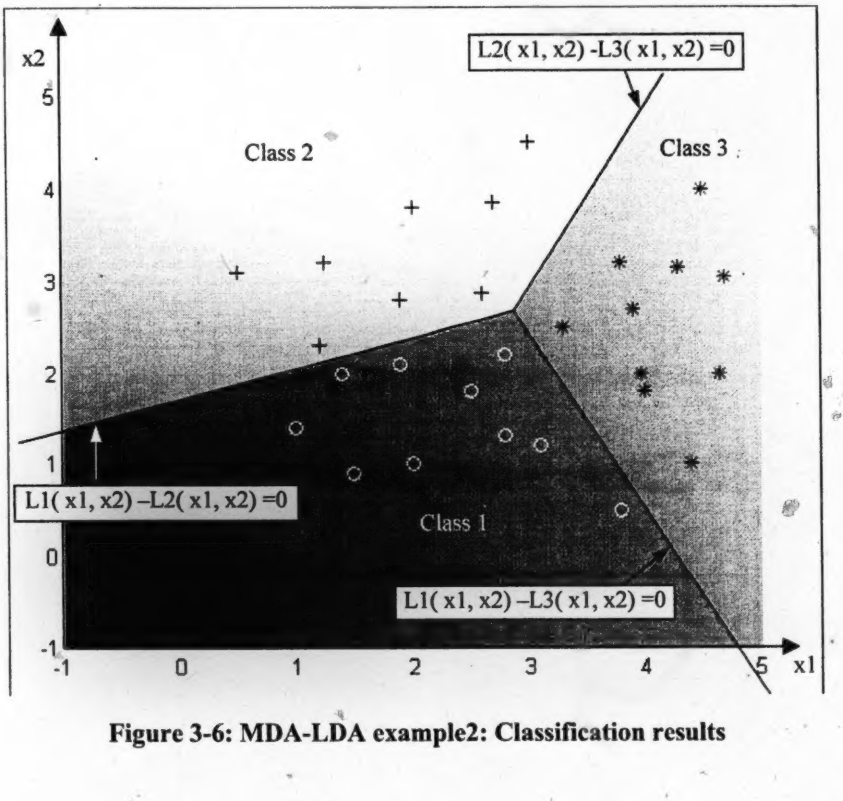


Figure 3-6: MDA-LDA example2: Classification results

3.3. ***Self Organizing Maps (SOM)***

The Self Organizing features Map (SOM) is a type of neural network introduced by Kohonen [38][39]. It performs unsupervised clustering in the form of vector quantization. It is a very valuable algorithm because it maps highly dimensional patterns (d-dimensional) onto a low dimensional space (usually 2-dimensional) while preserving the topology of the data.

The most commonly used configuration comprises a 2D array of M neurons, each one associated with a d-dimensional weight vector $\mathbf{w}_i=[w_{i1}, w_{i2},\ldots, w_{id}]$, as shown in Figure 3-7. The input vector $\mathbf{x}=[x_1, x_2,\ldots, x_d]$, is inputted in parallel to all the neurons of the map. After a selection process, the input is mapped to one of the neurons.

The kind of information obtained from the SOM is mostly qualitative. For example, when the training is finished, the system generates a map (usually a color or gray level map) that reports how close to the neuron weights the vectors mapped to each neuron are. The interpretation of this map could be done numerically, but it is the visualization tool that allows the user to obtain a better idea of how the data is distributed in the space.

The final values of the weights also give an insight of the structure of the data. These weights are generally stored in a matrix such as the one presented in Figure 3-9. The matrix contains as many layers as features the input data have. If taken separately, these layers generate the so-called weight maps that give a visual representation of the magnitudes of each feature on each neuron in the form of a simplified 2-dimensional histogram. If two weight maps are very similar, then the features they describe are correlated.

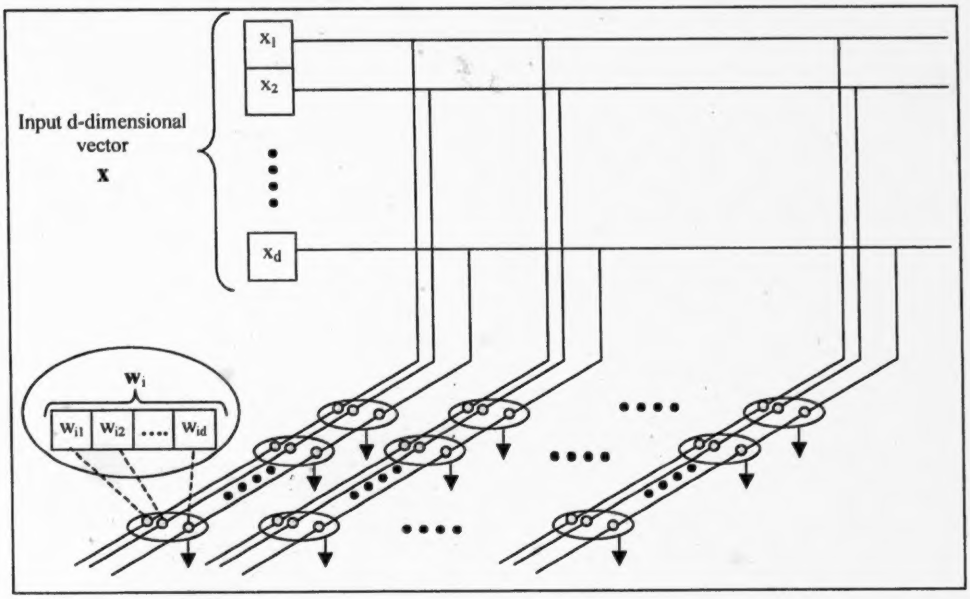


Figure 3-7: Two-dimensional self-organizing system

Both the learning process of the SOM and the interpretation of its results require the useful concept of neighborhood, which is described as follows:

The neighborhood of a neuron is the collection of neurons that lay on a certain radius δ around it. The "shape" of this surrounding area depends on how the neurons are arranged. In effect, different topologies produce different shapes on the definition of the neighborhood as shown in the examples of Figure 3-8. In the case of the square topology, a neuron has eight immediate neighbors, i.e. eight neurons are found on a radius δ=1 around a neuron. On the other hand, when the hexagonal topology is used, the immediate neighborhood is limited to six neurons. The advantages of using one topology over the other are not clearly defined although they clearly have a visual impact on the resulting map.

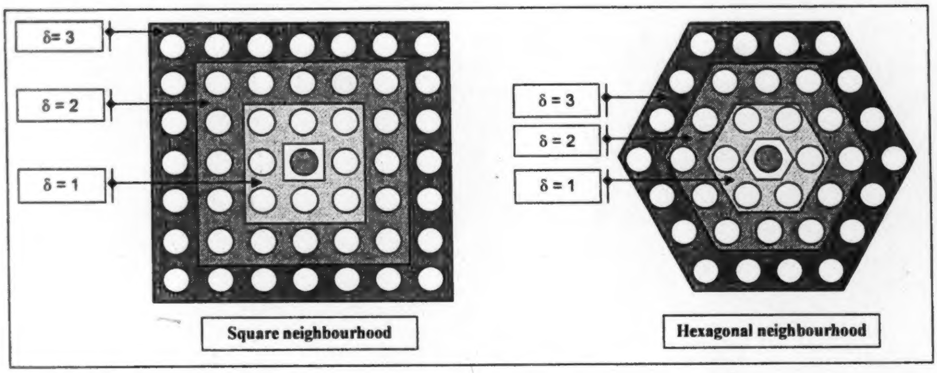


Figure 3-8: Radii assignation (δ) for two different neighborhood topologies: square and hexagonal

The SOM algorithm makes use of competitive learning for adjusting the weights of its neurons in an iterative process. Successive training samples are presented to the network, each of which alters the weights in the neurons. Once all the training samples have been presented to the net, one epoch has occurred. In short, two notions of timing are introduced: **one iteration** occurs each time an input sample is presented to the net, and **one epoch** is completed when all the training samples have been presented to the net.

During an iteration the network is faced to a d-dimensional input vector $\mathbf{x}=[x_1, x_2,\ldots, x_d]$, the neurons of the SOM compete against each other. In effect, either the distance or the dot product between **x** and **w** of each neuron is calculated and the input vector is mapped to the winning neuron j; that is, to the neuron from which the smallest distance or largest dot product is obtained respectively. This causes the weights of the winning neuron and its neighbors to be adjusted following the learning rule:

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \eta(k, te)(\mathbf{x} - \mathbf{w}_k(t)) \tag{3-19}$$

where

- t is the number of the present iteration, while t+1 is the number of the next iteration.
- te is, depending on the interpretation of the designer, the epoch in which the present iteration is or the actual iteration number.
- $\mathbf{w}_k(t)$ is the weight vector of a neuron k in the neighborhood of the winning neuron j on iteration (t); that is, before being adjusted.
- $\mathbf{w}_k(t+1)$ is the weight vector of a neuron k in the neighborhood of the winning neuron j on time (t+1); that is, after being adjusted.

- η (k,te) is the neighborhood kernel function that controls how much the weights of the neurons are being altered depending on the size of $N_j(te)$, the winning neighborhood at time te.

The neighborhood kernel η (k, j, te) has the form:

$$\eta(k, j, te) = \alpha(te) h\left(\left\| r_j - r_k \right\|, te\right) \tag{3-20}$$

where

- $\alpha(te)$ is a component of the kernel function that decreases monotonically with the time following an equation of the form $\alpha(te) = \alpha_{max} \times (1 - te/T)$, T being the expected total number of iterations.
- $\left\| r_j - r_k \right\|$ is the distance between the radius where the neurons j and k are on the grid
- $h\left(\left\| r_j - r_k \right\|, te\right)$ is a function that controls how much the weights of the neurons in the winning neighborhood $N_j(\delta(te))$ are to be adjusted; that is, the neurons inside $\delta(te)$ around the winning neuron are altered by a function defined by h. Note that $\delta(te)$ is the radius around the winning neuron where the neurons are considered to be in the winning neighborhood. δ starts relatively wide and then decreases monotonically with the time te.

On its simplest version, the function $h\left(\left\| r_j - r_k \right\|, te\right)$ takes the value one for all neurons k in the winning neighborhood and zero elsewhere, allowing $\alpha(te)$ to affect uniformly the winning neighborhood.

A more refined version includes the Gaussian function in the form:

$$h\left(\left\| r_j - r_k \right\|, te\right) = \exp\left(\frac{\left\| r_j - r_k \right\|^2}{2\delta^2(te)}\right) \tag{3-21}$$

in which case $\alpha(te)$ is scaled over all the neurons of the map in such way that the farther a neuron is from the winning neuron, the less affected its weights are by it.

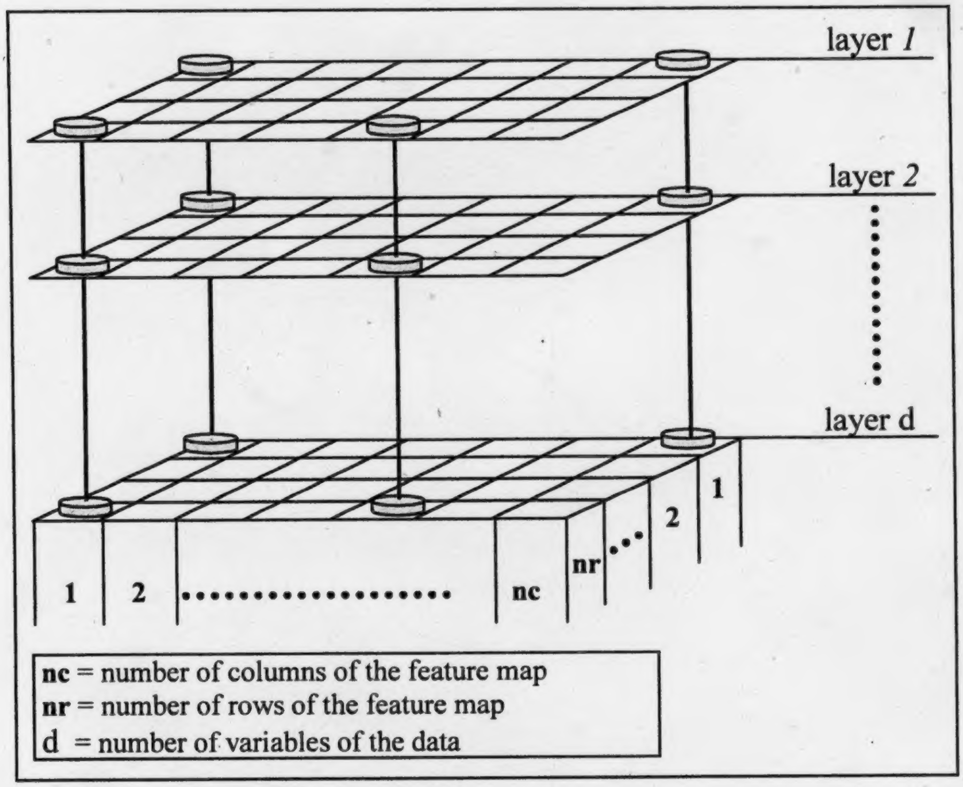

layer 1
layer 2
layer d
1
2
nc
nr
1
2
nc = number of columns of the feature map
nr = number of rows of the feature map
d = number of variables of the data

Figure 3-9: Configuration of the weights

The general algorithm for training the SOM is as follows:

GIVEN:

- topology of the network = 2D grid containing M neurons
- number of patterns of the training set = N
- number of features of each pattern = d
- number of training iterations = T.
- iteration number = t; epoch number = te (It could too be interpreted as iteration number)
- the neighborhood kernel = $\eta(k, j, te)$ as defined in (3-20)
- learning rate function = $\alpha(te) = \alpha_{max} \times (1 - te/T)$

The steps for completing the algorithm are the following:

1. initialize iteration step; t =0, te=0
2. initialize randomly the weights $\mathbf{w}_k(t)$, k=1, 2, …, M.

3. randomly order the input patterns of the training data set using a uniformly distributed probability density function.
4. initialize the index of the input pattern to one; $i = 1$.
5. present an input pattern $\mathbf{x}_i$ to the network
6. compute $\alpha(te)$ using the learning rate function corresponding to step te
7. compute either the Euclidean distances: $\| \mathbf{x}_i - \mathbf{w}_k(t) \|$, $k = 1, \ldots, M$

   or the dot products $\mathbf{y}_k = \mathbf{w}_k\, \mathbf{x}_i$, $k=1, \ldots, M$
8. select the j^{th} winning neuron as $\| \mathbf{x}_i - \mathbf{w}_j(t) \| = \min \| \mathbf{x}_i - \mathbf{w}_k(t) \|$, $k = 1, \ldots, M$

   or as:

   $$\mathbf{y}_j = \max(\mathbf{y}_k),\ k=1, \ldots, M$$
9. define the neighborhood kernel $\eta(k, j, te)$ around the winning neuron j
10. adjust the weights of the neurons using:

    $$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \eta(k, j, te)(\mathbf{x}_i - \mathbf{w}_k(t))$$

    or when using the dot product to select winning neuron j:

    $$\mathbf{w}_k(t+1) = \frac{(\mathbf{w}_k(t) + \eta(k, j, te)\, \mathbf{x}_i)}{(\| \mathbf{w}_k(t) + \eta(k, j, te)\, \mathbf{x}_i \|)}.$$

    Note that input and weight vectors have to be normalized when the winning neuron was chosen using the dot product.
11. increase $i = i + 1$. If $i > N$, reinitialize $i = 1$ and $te = te + 1$.
12. increase $t = t + 1$. If $t > T$ stop; otherwise go to step 5.

The previous algorithm permits the training of the SOM. After completing the training of the SOM, new patterns can be mapped on it by inputting them in parallel to all the neurons and determining the winning neuron.

Previously, the SOM has been referred to as an unsupervised clustering algorithm capable of mapping patterns from a highly dimensional space to a two-dimensional map, preserving its topology. This valuable characteristic could be exploited, among other things, to build a classifier, given that the dataset provides markers to know the position of the points of interest and corresponding classification labels. A simple way to do so is to assign a label to each cluster. In effect, it is very likely that similar patterns pertain to the same class. Once the structure of the data is found ,it makes sense to label the members of each cluster as pertaining to the dominant class of the cluster. When new patterns are mapped to the trained network, they receive the label of the cluster in which they fall. A cluster can be either a single neuron of the map or a group of them; given they are sufficiently close to each other. This is something the designer has to tune because it is application dependant. It is a matter of interpretation of the results obtained using the SOM.

An interesting study on how to select meaningful clusters from the SOM has been presented in the literature [80]. In that work, the authors used a SOM on the complete data set. They used its outputs to generate a reduced data set on which another unsupervised clustering process obtained the final clusters that were not single neurons of the Map. Another way to group neurons of the Map and make more general clusters is using visual inspection and grouping the adjacent neurons that have similar color. One more alternative to finding clusters in the SOM is to use the classification labels of the points of interest in the data and group the neurons based on entropy.

3.4. Radial Basis Functions Neural Networks (RBF NN)

Radial Basis Functions (RBF) appeared as an approach for performing strict interpolation in a multidimensional space [62]. In the case of a d-dimensional to one-dimensional space mapping, it is intended to find a function h that permits the conversion of a set of N d-dimensional vectors $\mathbf{x}^n$ to their corresponding target value t^n. That is:

$$h(\mathbf{x}^n) = t^n \text{ , for } n=1,\ldots,N \tag{3-22}$$

The function h(x) is defined as the weighted sum of N basis functions $\phi(\|x - x^n\|)$ and has the general form:

$$h(\mathbf{x}) = \sum_{n=1}^{N} w_n \phi(\| \mathbf{x} - \mathbf{x}^n \|), \tag{3-23}$$

where the non-linear function $\phi(\cdot)$ is applied to the distance from an input vector $\mathbf{x}$ to a prototype $\mathbf{x}^n$.

Once the configuration of the basis function is decided, it is necessary to calculate the corresponding weights w_n.

This can be accomplished by writing equation (3-22) in matrix form as:

$$\mathbf{\Phi w} = \mathbf{t} \tag{3-24}$$

where $\mathbf{t} \equiv (t^n)$, $\mathbf{w} \equiv (w_n)$, and $\mathbf{\Phi}$ is a square matrix that has elements $\Phi_{nn'} = \phi(\|x - x^n\|)$

Then, equation (3-24) can be solved with $\mathbf{w} \equiv \mathbf{\Phi}^{-1}\mathbf{t}$ if $\mathbf{\Phi}^{-1}$ exists.

For the actual implementation of h(**x**), there is no unique type of distance or of non-linear function to form the basis function. Table 3-1 groups some of the non-linear functions mentioned in the literature [6], the Gaussian being the most commonly used for its useful analytical properties.

Table 3-1: Several forms of basis functions considered in the lecture.

Gaussian	$\phi(v) = \exp\left(-\frac{v^2}{2\sigma^2}\right)$ (3-25) Where σ is a real value that controls how smooth the function is. It gives a localized basis function: $\phi \to 0$ as $\lvert v \rvert \to \infty$
Thin-plate spline	$\phi(v) = v^2 \ln(v)$
quadratic	$\phi(v) = \left(v^2 + \sigma^2\right)^{\beta}$ called Multi-quadric when $\beta=1/2$ and inverse Multi-quadric when $\beta= -1/2$
Cubic	$\phi(v) = v^3$
Linear	$\phi(v) = v$. For the basis function, the non-linearity is all due in this case to the distance-to-the-prototype measurement.

The type of distance employed depends on the perception the user has of the data structure. There are several available measures of distances also called indices of proximity presented in the literature [16]; these are summarized in Table 3-2. The type of distance used alters the location of the family of vectors that are considered to be at the same distance from a reference vector. This effect is illustrated for the 2-dimensional case in Figure 3-10. In the case of the Euclidean distance, this family forms a circle, a sphere or a hyper sphere around the reference vector depending if the vectors are two dimensional, three dimensional or highly dimensional. In the same way, in the two dimensional case, the Hamming distance produces a diamond shape around the reference, whereas Tschebyschev distance generates a square. Mahalanobis distance adjusts this shape to reflect the relationship among the variances of the variables. In effect, it generates d-dimensional rotated ellipses. As a special case of the Mahalanobis metric, the Bhattaharaya distance gives rise to non-rotated d-dimensional ellipses only taking into account the variance of each variable.

Table 3-2: Type of distance measures

Minkowsky metric	$d(\mathbf{x}_q,\mathbf{x}_r)=\left(\sum_{j=1}^{d}\left\|x_{qj}-x_{rj}\right\|^{m}\right)^{1/m}$	d is the number of features of vectors $\mathbf{x}_q$ and $\mathbf{x}_r$.
Euclidean distance	$d(\mathbf{x}_q,\mathbf{x}_r)=\left(\sum_{j=1}^{d}\left\|x_{qj}-x_{rj}\right\|^{2}\right)^{1/2}$	Minkowski metric with m=2
Manhattan or Hamming distance	$d(\mathbf{x}_q,\mathbf{x}_r)=\left(\sum_{j=1}^{d}\left\|x_{qj}-x_{rj}\right\|\right)$	Minkowski metric with m =1
Tschebyschev distance	$d(\mathbf{x}_q,\mathbf{x}_r)=\max\left(\left\|x_{qj}-x_{rj}\right\|\right)_{i=1,2,\ldots,d}$	Minkowski metric with m =∞
Mahalanobis distance	$d(\mathbf{x}_q,\mathbf{x}_r)=(x_q-x_r)^{T}\mathbf{\Sigma}^{-1}(x_q-x_r)$	$\mathbf{\Sigma}$ is the covariance matrix.
Bhattaharaya or weighted Euclidean distance	$d(\mathbf{x}_q,\mathbf{x}_r)=\left(\sum_{j=1}^{d}\frac{(x_{qj}-x_{rj})^{2}}{\sigma_j^{2}}\right)$	Mahalanobis distance when $\mathbf{\Sigma}$ is diagonal. $(\sigma_j)^2$ is the variance of variable j.

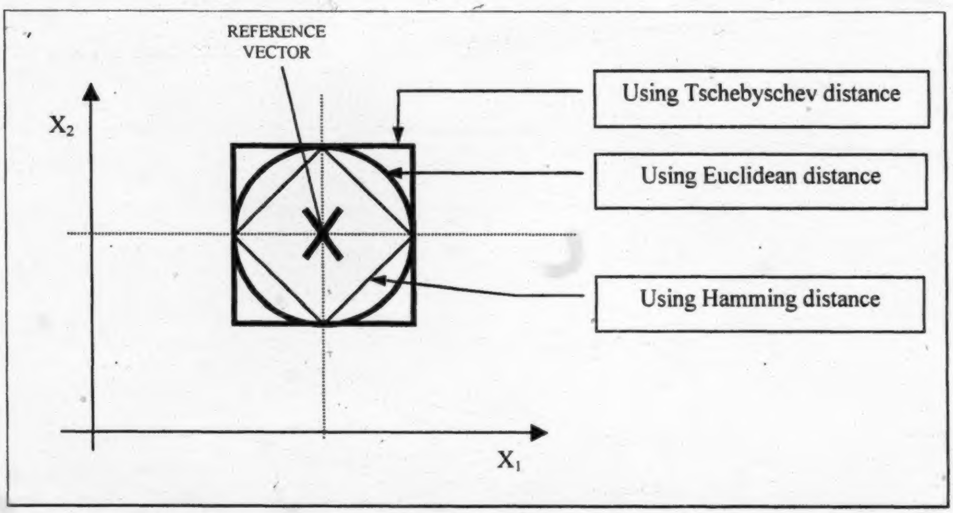


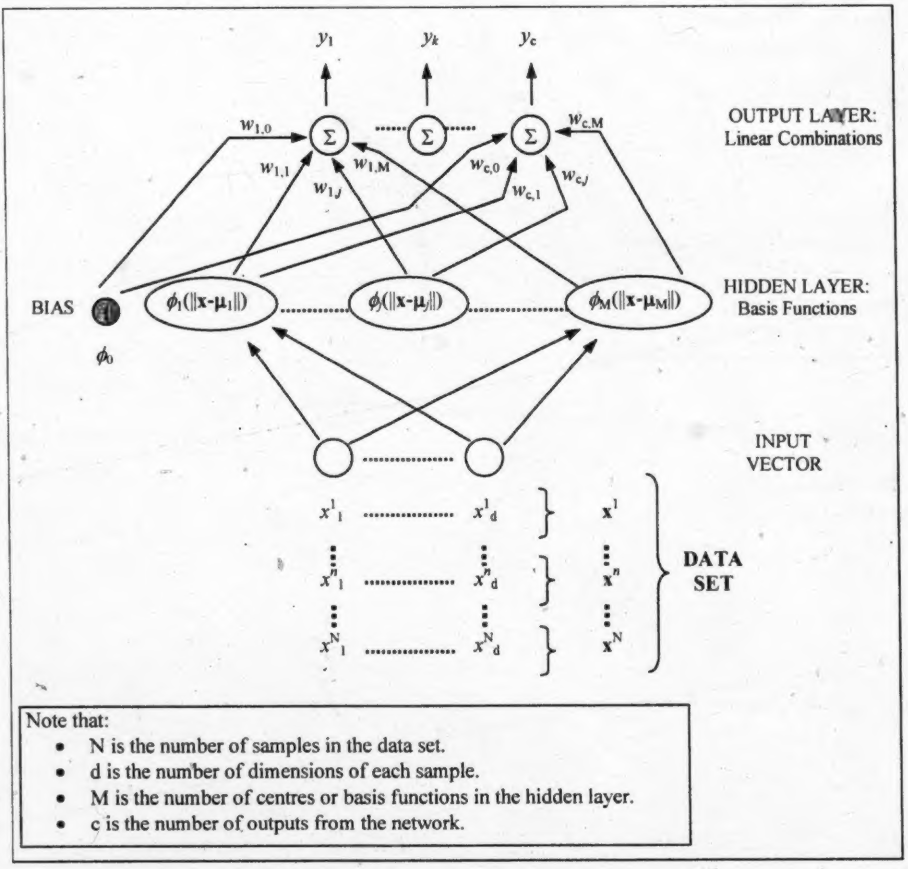
Figure 3-10: Location of the vectors equidistant to a reference vector using Euclidean, Hamming and Tschebyschev distances; example with two variables

The exact interpolation basis functions gave birth to the RBF network from which a smoother result is obtained product of the reduction of the number of basis function. In effect, the number M of centers, and consequently the number of basis functions in the network is much less than the number of data samples N [6]. The training process of this network would help select suitable centers μ_j as well as the width σ of each basis function. To satisfy these characteristics, equation (3-23) gives place to:

$$y_k(\mathbf{x}) = \sum_{j=0}^{M} w_{kj}\, \phi_j\left(\left\|\mathbf{x} - \boldsymbol{\mu}_j\right\|\right)$$

(3-26)

As shown in Figure 3-11, the input vector goes to a hidden layer composed by M basis functions for a non linear transformation of the distance between the input vector **x** and the center μ. The contribution of each basis function is then weighted and added to a bias weight to obtain the output at the so-called linear part of the network.


y_1
y_k
y_c
$w_{1,0}$
$w_{c,M}$
OUTPUT LAYER: Linear Combinations
$w_{1,1}$
$w_{1,M}$
$w_{1,j}$
$w_{c,0}$
$w_{c,j}$
$w_{c,1}$
BIAS
ϕ_0
$\phi_1(\|\mathbf{x}-\mu_1\|)$
$\phi_j(\|\mathbf{x}-\mu_j\|)$
$\phi_M(\|\mathbf{x}-\mu_M\|)$
HIDDEN LAYER: Basis Functions
INPUT VECTOR
x^1_1 x^1_d $\mathbf{x}^1$
x^n_1 x^n_d $\mathbf{x}^n$
x^N_1 x^N_d $\mathbf{x}^N$
DATA SET

Note that:

- N is the number of samples in the data set.
- d is the number of dimensions of each sample.
- M is the number of centres or basis functions in the hidden layer.
- c is the number of outputs from the network.

Figure 3-11: Structure of the Radial Basis Functions Network

The training process of the RBF networks can be executed in two consecutive steps: The first step consists of learning, from the data structure, the parameters of the hidden layer.

The second step consists in finding the values of the weights of the linear combiners at the output layer, while the basis functions are kept fixed.

The second step of training is common to all the possible configurations that could be obtained on the first step. It will hence be described first.

The values of the weights are calculated by minimizing the error function:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{c}\left\{\sum_{j=1}^{M} w_{kj}\phi_j^n - t_k^n\right\}^2 \tag{3-27}$$

where t_k^n is the target value of the kth class for the nth input sample.

Setting to zero the derivative of equation (3-27) with respect to w_{kj} the expression is minimized and it leads to the following equation [6]:

$$\left(\mathbf{\Phi}^{T}\mathbf{\Phi}\right)\mathbf{W}^{T} = \mathbf{\Phi}^{T}\,\mathbf{T} \tag{3-28}$$

where **T** is the (N x c) target matrix, **W** is the (c x M) matrix of output weights in the form w_{kj}, **Φ** is the (N x M) matrix resulting from applying the M basis functions of the hidden layer to the N input samples. Elements of **Φ** have the form ϕ_j^n or $\phi_j\left(\mathbf{x}^n\right)$.

The weights can be obtained from (3-28) using linear matrix inversion techniques [26][64] leading to the formal solution:

$$\mathbf{W}^{T} = \mathbf{\Phi}^{\dagger}\mathbf{T} \tag{3-29}$$

where $\mathbf{\Phi}^{\dagger} \equiv \left(\mathbf{\Phi}^{T}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{T}$ is called the pseudo inverse of **Φ**

Back to the first step of training, there are several approaches to making sense of the structure of the data and to finding the centers and configurations of the basis functions that suit it. One early approach was to arbitrarily select centers from the training data set until a certain criterion was satisfied. This approach was likely to lead to unnecessarily large RBF networks without providing the expected efficiency. With the purpose of providing a more efficient solution to this problem, a learning algorithm based on Orthogonal Least Squares (OLS) was introduced in [9].

The training of the OLS is iterative. It consists of evaluating the error of the RBF over the whole data set when using each sample of the data set as a center, one at the time. The value of the weights of the second layer is calculated for each time. The sample that, when used as a center, produces the minimal error becomes a new center of the hidden layer of the network before the next center selection iteration. The process lasts until a maximum number of samples of the data set has been selected as centers or the error of the RBF configuration reaches an acceptable level, whatever comes first. The generalization capability of this network depends on the value chosen to be an acceptable error. The details of an efficient algorithm for achieving the described result in a more efficient way can be found in [9].

With the previously mentioned techniques, the centers were chosen from the data set. These centers may not bring the best generalization capacity to the network. For example, Figure 3-12 illustrates a subset of a determined data set. Samples are relatively near one another and comparatively far from the rest of the data set. When one center that represents this subset of the data is sought by OLS, either s(1) or s(2) is chosen, but one may intuitively find that it is μ that describes more accurately this subset.

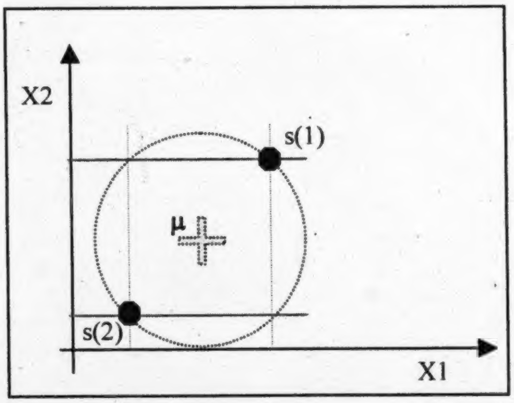


Figure 3-12: Subset of a data set formed by 2-dimensional vectors.

As the best centers are not always found among the samples of the training data set, the motivation for the use of clustering algorithms arises. Algorithms such as the K-means clustering algorithm [52] and the Self Organizing Map of features (SOM) [38][39] bring unsupervised methods for generating the clusters and/or the centers of each cluster.

K-means clustering consists in an iterative procedure where the training data set is partitioned into K subsets. The mean vector μ_j of each subset is called the prototype of subset j. After the prototype is calculated, the data set is redistributed so that each sample is assigned to the subset with the nearest prototype. This process of calculating the mean vector of the subsets and then redistributing the samples among the subsets is repeated until the changes on the values of the prototypes from one iteration to the next one is sufficiently small. This algorithm seeks to minimize the total distance from the members of each subset S_j to its prototype $\boldsymbol{\mu}_j$ expressed as:

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} \left\| \mathbf{x}^n - \boldsymbol{\mu}_j \right\|^2$$

(3-30)

where J is the sum-of-squares clustering function, K is the number of clusters, and $\boldsymbol{\mu}_j$ is the mean of the samples in S_j

$$\boldsymbol{\mu}_j = \frac{1}{N_j} \sum_{n \in S_j} \mathbf{x}^n$$

(3-31)

Using the K-means algorithm to train the hidden layer of the RBF implies including a method to determine a suitable value for the number of centers K. This value could be chosen similarly to with the OLS method as the smallest value of K that generates a sufficiently small error at the output.

Clustering with SOM has been previously described in Section 3.3 and the details on its use with the RBF will be explored more fully in the next Section.

At this point, both the way to calculate the weights of the output layer and several ways to estimate the centers of the hidden layer have been described. The problem is that not only the centers of the hidden layers have to be determined but also the parameters that make the basis functions fit the structure of the data around those centers. In the case of the Gaussian basis function shown in equation (3-25), this parameter is the width measure σ. The methods vary depending on the application. The width σ_j of the subset j basis function could be estimated as a multiple of the average distance from the prototype of the subset ($\boldsymbol{\mu}_j$) to the rest of the centers. Another alternative is to choose the width σ_j to be the average distance from the center j to its L nearest neighbors. One can also examine the resulting clusters as the number of clusters increase and seek a stopping rule. Several indexes serving this purpose are presented in [16].

Note that σ_j can take either a unique value for all the features of the center or different values for each feature according to its distribution. It can even be generalized to an arbitrary covariance matrix $\mathbf{\Sigma}$ converting (and adapting) equation (3-25) to:

$$\phi_j(\mathbf{x}) = \exp\left\{-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}_j\right)^T \mathbf{\Sigma}_j^{-1}\left(\mathbf{x}-\boldsymbol{\mu}_j\right)\right\}$$

(3-32)

where the expression $\left\{\left(\mathbf{x}-\boldsymbol{\mu}_j\right)^T \mathbf{\Sigma}_j^{-1}\left(\mathbf{x}-\boldsymbol{\mu}_j\right)\right\}$ is known as the Mahalanobis distance from $\mathbf{x}$ to $\boldsymbol{\mu}_j$. This option is accompanied with the increment on the number of adjustable parameters.

In summary, training the RBF network includes finding:

1. the number of centers
2. the actual centers
3. the values of the adjustable parameters of the basis functions pertaining to each center
4. the weights of the linear part of the RBF.

It is the designer's task to select the techniques and configurations to suit his or her problem.

3.5. Proposed hybrid RBF neural networks for classification purposes

The hybrid RBF neural network has the same basic structure as the RBF neural network presented in Section 3.4, and depicted in Figure 3-11.

The hybrid component is considered for the determination of the number of centers and their values.

The method to determine the centers that involve orthogonal least squares has two main drawbacks:

1. The value of the centers is limited to the values of the samples available in the training set. This means that the selected centers might not be really in the center of the zone they represent.
2. The RBF neural network is trained to obtain all its parameters each time that one center is added. This procedure involves an increasing amount of computation.

An alternative to the orthogonal least squares method is presented in this work to choose the centers of the radial basis functions by performing two-stage clustering using self organizing maps.

The first stage involves the unsupervised training of the SOM as presented in Section 3.3. The training data set is inputted to the SOM to be clustered. Certain heuristics have to be tried out for the training parameters, such as the number of epochs or iterations to complete the algorithm, the learning rate, the shape of the neurons, the characteristics of the neighborhood kernel, etc. The first stage is summarized in Figure 3-13. The output of the clustering process consists of:

1. the final weights assigned to each neuron of the two-dimensional array to which the input data is mapped. There is one weight value per variable in the data per neuron in the map. These weights can be visualized in the Weight Maps. There is a weight map per variable. For each neuron, it shows the magnitude of the weight assigned to the determined variable. This magnitude can be converted to gray levels to enable a graphical interpretation.
2. the degree of similarity of the samples mapped to each neuron, presented by the Clustering Map (**CM**) in the form of a matrix. Each one of these values can be expressed as a value of brightness (an integer between 0 and 255, where 0 is the darkest gray level corresponding to the less similar samples and 255 is the brightest gray level corresponding to the most similar samples). This type of information can be shown graphically for the user to choose neurons to be merged as clusters or to be used individually as clusters. An automatic interpretation can also be made based on this information using some rules. This case is discussed in the second stage of the process.
3. the relative amount hits (i.e. the amount of input patterns mapped to each neuron), presented by the Distribution Map (DM). This value can be also expressed in terms of gray levels. The highest value (255, the brightest gray level) represents the neuron with the lesser hits, while the lowest value (0, the darkest gray level) represents the neurons with the most hits.
4. the list of samples mapped to each neuron.

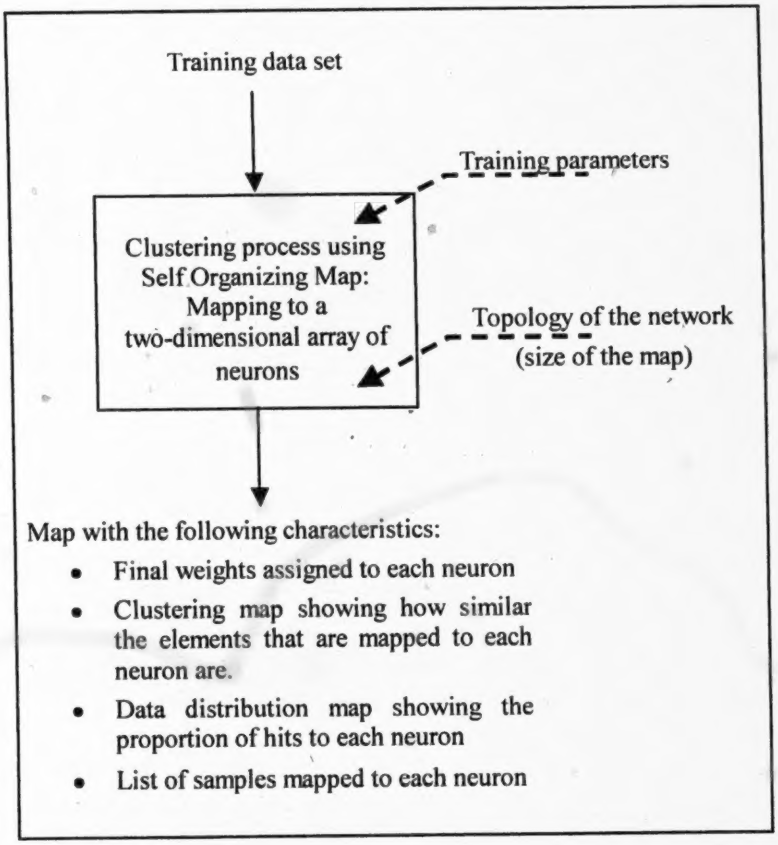
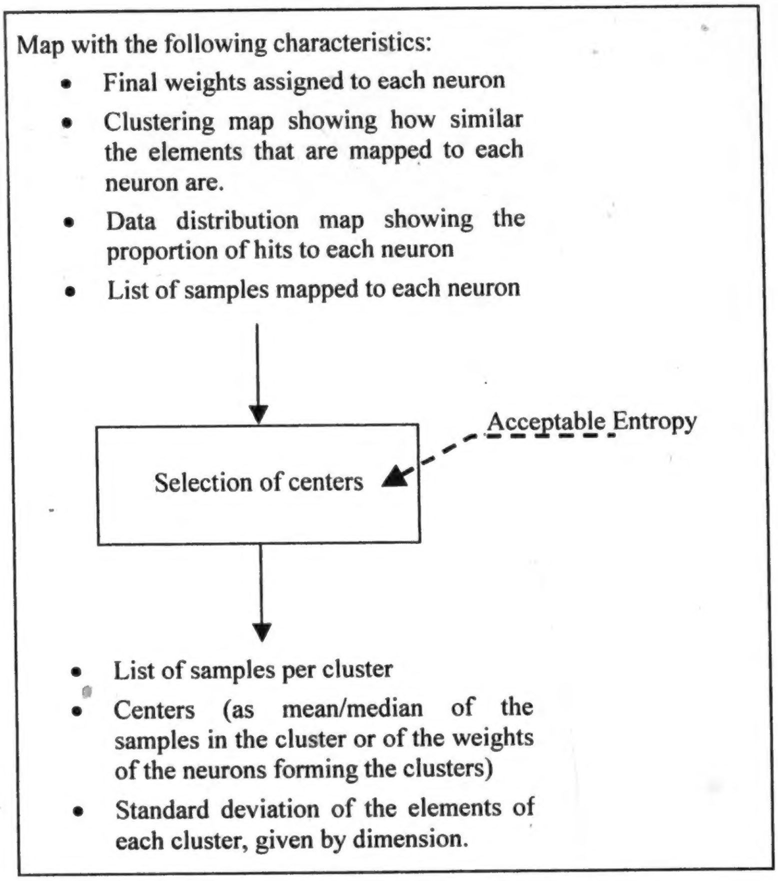

Training data set
Training parameters
Clustering process using Self Organizing Map: Mapping to a two-dimensional array of neurons
Topology of the network (size of the map)
Map with the following characteristics:
Final weights assigned to each neuron
Clustering map showing how similar the elements that are mapped to each neuron are.
Data distribution map showing the proportion of hits to each neuron
List of samples mapped to each neuron

Figure 3-13: Inputs/Outputs of the clustering process using SOM

In the second stage, the results obtained with the SOM algorithm are interpreted in order to satisfy not only the spatial distribution of the samples but also the distribution of the classes to which these samples belong.

The determination of the final clusters is based on: 1)the clustering map (**CM**), which gives the degree of similarity between the elements mapped to the neurons and 2) the entropy found among the elements of these clusters. The clusters obtained from step 1 are grouped to satisfy an acceptable value of entropy to determine how heterogeneous the distribution of classes is in a selected group of samples. This process is summarized in Figure 3-14.



Map with the following characteristics:

- Final weights assigned to each neuron
- Clustering map showing how similar the elements that are mapped to each neuron are.
- Data distribution map showing the proportion of hits to each neuron
- List of samples mapped to each neuron

- List of samples per cluster
- Centers (as mean/median of the samples in the cluster or of the weights of the neurons forming the clusters)
- Standard deviation of the elements of each cluster, given by dimension.

Figure 3-14: Inputs/Outputs of step 2

A detailed explanation on the procedure to automatically determine the clusters from the output of the SOM is depicted in Figure 3-15. A matrix arbitrarily called Matrix of Cluster Definitions (**MCD**) of the same size than the clustering map is created to assign a cluster number to each neuron. All its cells are initialized to (-1), meaning that none of the corresponding neurons belong to any cluster. An initial threshold is set on the similarity measures given by the clustering map. All the elements of the **MCD** corresponding to where the values of **CM** are above this threshold are set to (0), meaning that these neurons belong to a cluster yet to be assigned. The groups of (0) surrounded by (-1) are assigned a cluster number. In other words, each separated group of (0) will get a sequentially obtained cluster number.

In analyzing entropy, the entropy of the samples that were mapped to each group of neurons is first calculated. The clusters with acceptable entropy are maintained. The threshold of brightness for clusters with deficient entropy is increased, and the new clusters are obtained.

This process is repeated until there are no more neurons in the cluster or the entropy of the clusters finally becomes acceptable.

Finally, the neurons that could not be assigned to any cluster (all the neurons with (-1) in the **MCD**) can finally be analyzed one by one. If they satisfy the entropy criterion, if they have enough samples mapped to them (defined by the designer), and if they satisfy a minimum of similarity (also given by the designer), they become clusters themselves.

The raw outputs from this second stage are:

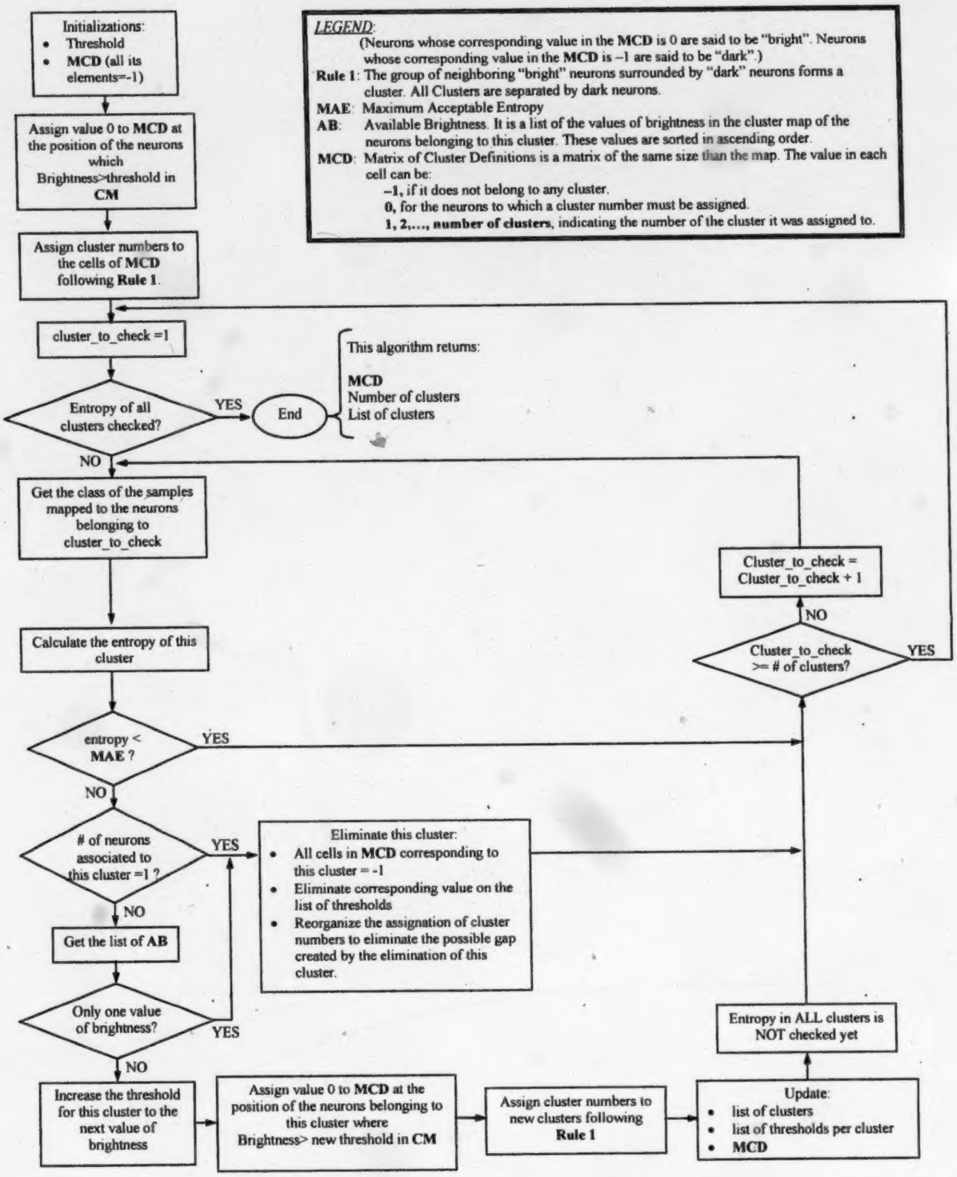
- the cluster assignation to each neuron (**MCD**);
- the list of thresholds used to obtain these clusters; and
- the list of samples per neuron, and indirectly the list of samples per cluster.

From these raw outputs, one can calculate the statistics of each cluster that will determine the values of the centers used by the RBF to define the centers of their basis functions. There are many options to determine these centers. The most suitable one should be selected after the experimental study. In effect the center of a cluster can be obtained as:

- the mean or the median of all samples belonging to the cluster. Given a multidimensional set of samples, the center would be formed by the means or the medians of these samples for each one of their features.
- the mean or the median of the weights that characterize all the neurons belonging to the cluster. For example, if three neurons were merged to form a cluster, then it is the mean or the median of their weights that is taken as the center of the cluster.

In order to train the RBF, it is also necessary to have at least an initial value of spread for each basis function. This spread can be also estimated from the information obtained in the two-steps clustering process. One option is to define it as a multiple of the standard deviation of the samples around the center of the cluster.

In summary, this process allows the designer to get a description of the clusters based on their members. It is up to the designer to interpret this information to get a better result.


Initializations:
Threshold
MCD (all its elements=-1)
Assign value 0 to MCD at the position of the neurons which Brightness>threshold in CM
Assign cluster numbers to the cells of MCD following Rule 1.
cluster_to_check =1
Entropy of all clusters checked?
YES
End
This algorithm returns:
MCD
Number of clusters
List of clusters
NO
Get the class of the samples mapped to the neurons belonging to cluster_to_check
Calculate the entropy of this cluster
entropy < MAE ?
YES
NO
# of neurons associated to this cluster =1 ?
YES
Eliminate this cluster:
All cells in MCD corresponding to this cluster = -1
Eliminate corresponding value on the list of thresholds
Reorganize the assignation of cluster numbers to eliminate the possible gap created by the elimination of this cluster.
NO
Get the list of AB
Only one value of brightness?
YES
NO
Increase the threshold for this cluster to the next value of brightness
Assign value 0 to MCD at the position of the neurons belonging to this cluster where Brightness> new threshold in CM
Assign cluster numbers to new clusters following Rule 1
Update:
list of clusters
list of thresholds per cluster
MCD
Entropy in ALL clusters is NOT checked yet
Cluster_to_check >= # of clusters?
YES
NO
Cluster_to_check = Cluster_to_check + 1

LEGEND:

(Neurons whose corresponding value in the **MCD** is 0 are said to be "bright". Neurons whose corresponding value in the **MCD** is –1 are said to be "dark".)

Rule 1: The group of neighboring "bright" neurons surrounded by "dark" neurons forms a cluster. All Clusters are separated by dark neurons.

MAE: Maximum Acceptable Entropy

AB: Available Brightness. It is a list of the values of brightness in the cluster map of the neurons belonging to this cluster. These values are sorted in ascending order.

MCD: Matrix of Cluster Definitions is a matrix of the same size than the map. The value in each cell can be:

–1, if it does not belong to any cluster.

0, for the neurons to which a cluster number must be assigned.

1, 2,..., number of clusters, indicating the number of the cluster it was assigned to.

Figure 3-15: Clustering of the SOM based on entropy

4. Experimental Study

4.1. Introduction

The experiments performed in this study are shown in Figure 4-1. This work was divided into three main parts:

1. Feature extraction and selection
2. QRS detection.
3. Classification of the QRS (class N, V, R, and L):
    a) using Linear Discriminant Analysis (LDA)
    b) adapting the Self Organizing Map Neural (SOM) networks to classify the beats.
    c) using Orthogonal Least Squares Radial Basis Functions (RBF-OLS) neural networks.
    d) using Radial Basis Functions for which the centers are obtained using SOM clustering (SOM-RBF).

For the QRS detection three of the four methods explained in Section 2.5.3 were explored: the digital filter by Okada [55], the MOBD [77], and a neural network [21]. An improved version of the Okada algorithm was also implemented.

The experiments were done on the data set described in Section 2.6.2. Each method of QRS detection resulted in a number of correct detections (true positives) and incorrect detections (false positives and false negatives) per record and in average foe all the records.

The experiments related to the feature extraction consisted of two different methods to reduce the number of features of the data: performing the Principal Component Analysis (PCA), and transforming each QRS segment into a Hermite series of expansion. It will be explained in the methodology of each experiment that the selection of the number of features was based on the results of:

1. the calculations of squared error and correlation index between the original signal and the modeled one;
2. the results when using LDA classification on the selected data set; and
3. in the PCA case, the changes in the variance due to the number of principal components chosen.

The LDA classification method was explained in Section 3.2. The classification experiments involving LDA are summarized briefly in this chapter because they are presented in the same Section as the feature extraction and selection to provide a notion of how far or close the different classes are in the new feature space. The LDA experiments are done on the

QRS segments, on their principal components, and on their transformation to Hermite coefficients.

For the experiments applying SOM, RBF-OLS and SOM-RBF, the data set of QRS segments and the Hermite coefficients data were used. The data was either normalized using logistic normalization or not normalized at all.

The use of the SOM as a classifier was explained in Section 3.3. The experiments consisted in trying different sizes of maps on the different data sets. Each neuron was labeled as pertaining to the class to which the majority of the samples in it pertain.

The RBF Neural Networks was explained in Section 3.4. The OLS method to complete this classifier was explained at the end of the same Section.

The experiments regarding the SOM-RBF classifier include numerous options. In effect, these experiments involve the different sizes of the SOM maps, the criteria to select the clusters in the SOM maps, the criteria to select the prototype of each of these clusters, and the criteria to select the initial spread of the RBF functions. They will be explained in detail in the methodology of the corresponding experiments.

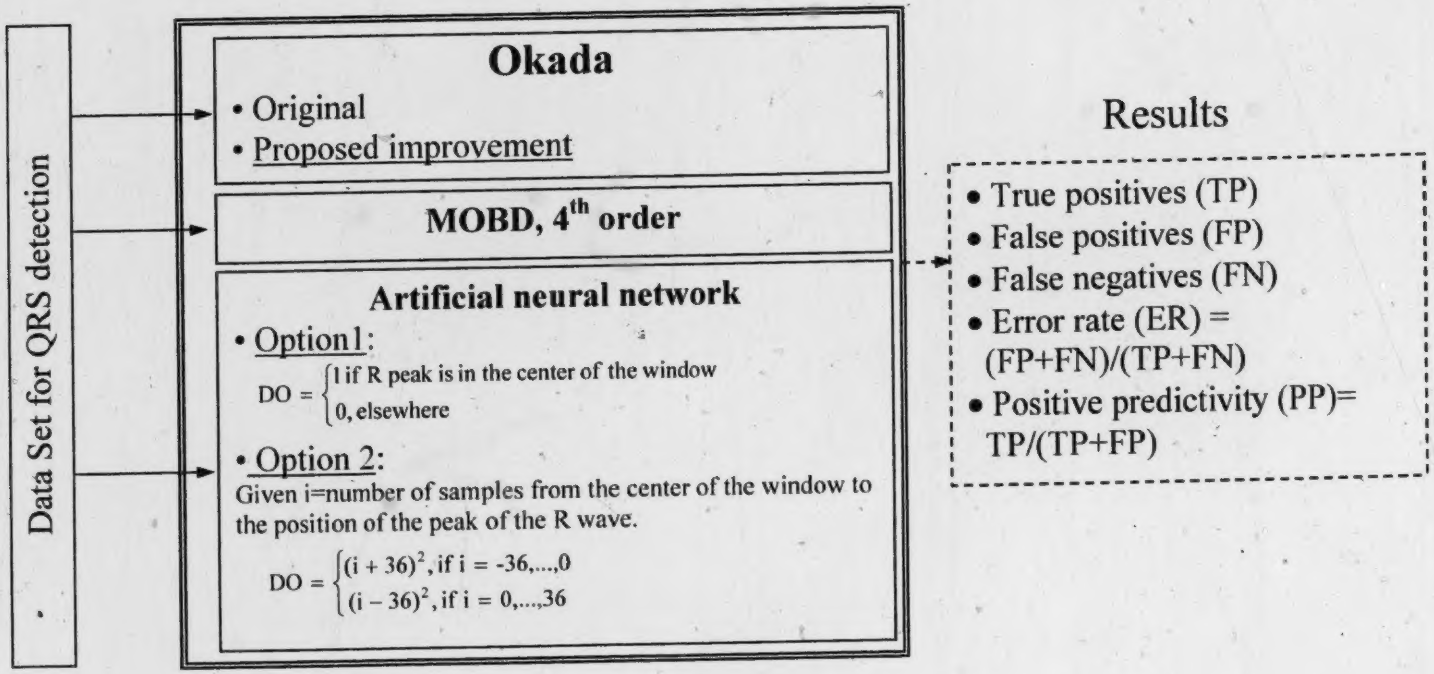

Data Set for QRS detection
Okada
• Original
• Proposed improvement
MOBD, 4th order
Artificial neural network
• Option1:
$DO = \begin{cases} 1 \text{ if R peak is in the center of the window} \\ 0, \text{elsewhere} \end{cases}$
• Option 2:
Given i=number of samples from the center of the window to the position of the peak of the R wave.
$DO = \begin{cases} (i + 36)^2, \text{if } i = -36,...,0 \\ (i - 36)^2, \text{if } i = 0,...,36 \end{cases}$
Results
• True positives (TP)
• False positives (FP)
• False negatives (FN)
• Error rate (ER) = (FP+FN)/(TP+FN)
• Positive predictivity (PP)= TP/(TP+FP)

Figure 4-1: Experimental work

4.2. Feature extraction and selection

In this Section, the results of the study concerning the extraction of features using PCA and Hermite Transformation are shown.

The study includes calculations of several measures of fit described in Section 2.4.5. Based on the results obtained, the final number of features selected for the PCA is shown in Section 4.2.1 and for the Hermite transformation in Section 4.2.2.

4.2.1. Feature extraction and selection using PCA

The Principal Component Analysis is used to reduce the number of variables of the original data set. This is achieved by converting the data set characterized by O variables to a dataset characterized by d features as explained in Section 2.5.2.C. As previously cited in Section 2.6.3, the original data set has O= 75 variables (73 samples of the ECG signal –specifically the QRS segments- plus two values of local rhythm).

The purpose of this experiment was to estimate the number of principal components to use to represent appropriately the original the data in study.

Using the data set noted in Section 2.6.4, the experiment consisted in performing the Karhunen-Loeve transformation explained in Section 2.5.2.C to obtain the principal components. The selection of the number of principal components was based on four series of calculations:

- the contribution of each principal component to the total variance.
- the correlation of the original data set to its approximation using the inverse Karhunen-Loeve transformation with different number of principal components.
- the squared error of the approximation with respect to the original data set.
- the accuracy of the LDA classifier for different number of principal components.

A. Contribution to the total variance

The fraction of variance is calculated for each one of the possible 75 principal components to see how much they contribute to the total variance of the data set. To have a better visualization, the cumulative contribution is also presented. The results are shown in Figure 4-2 and Figure 4-3.

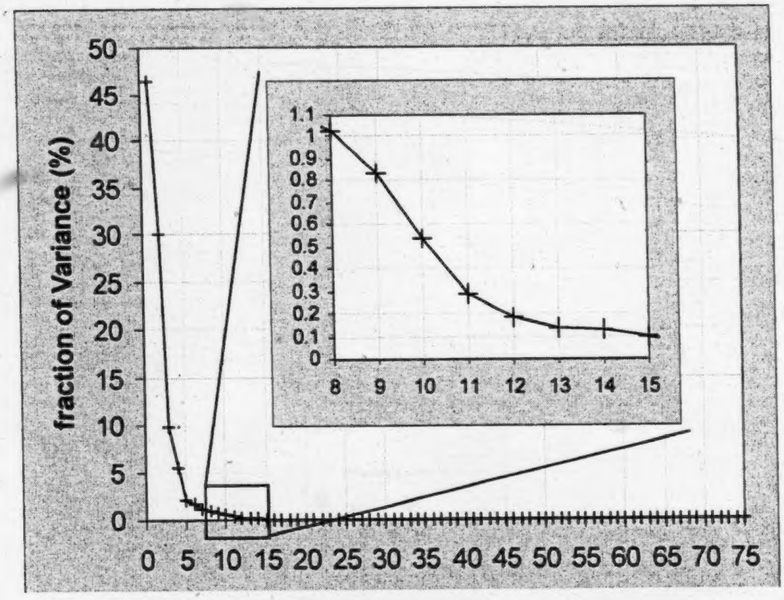


Figure 4-2: % Fraction of Variance contributed by each principal component in the PCA

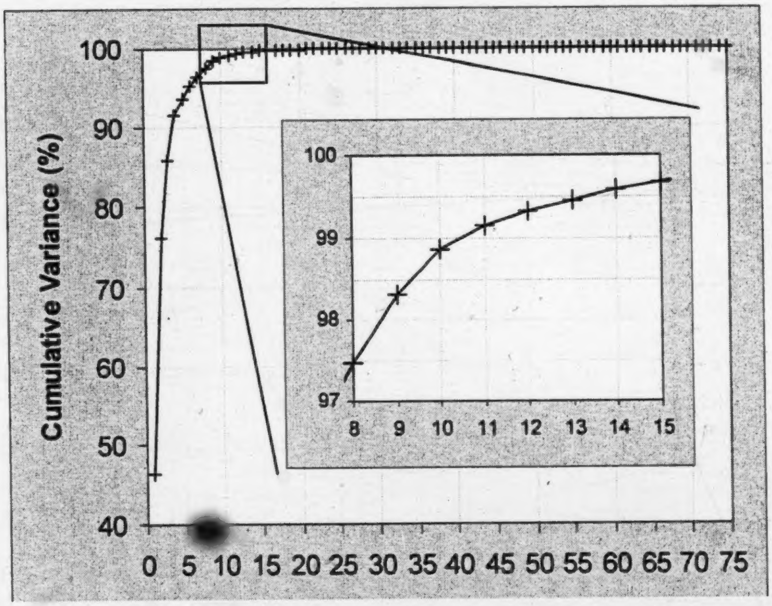


Figure 4-3: Cumulative Variance when using different number of principal components for the PCA

From the results presented above, it can be said that the first ten principal components contribute highly to the total variance of the data set. From this point, the contribution declines and then, from the fifteenth principal component on, the contribution is minimal.

B. Squared error and correlation index

Once the Karhunen-Loeve transformation was performed on the original data set, its inverse transformation was performed to obtain an approximation of the original data set. This inverse transformation was performed 75 times to obtain 75 approximations using a different number of principal components.

The squared error (explained in Section 2.4.5.A) and the correlation index (explained in Section 2.4.5.B) between both versions were calculated for each approximation. The results are shown in Figure 4-4 and in Figure 4-5 respectively.

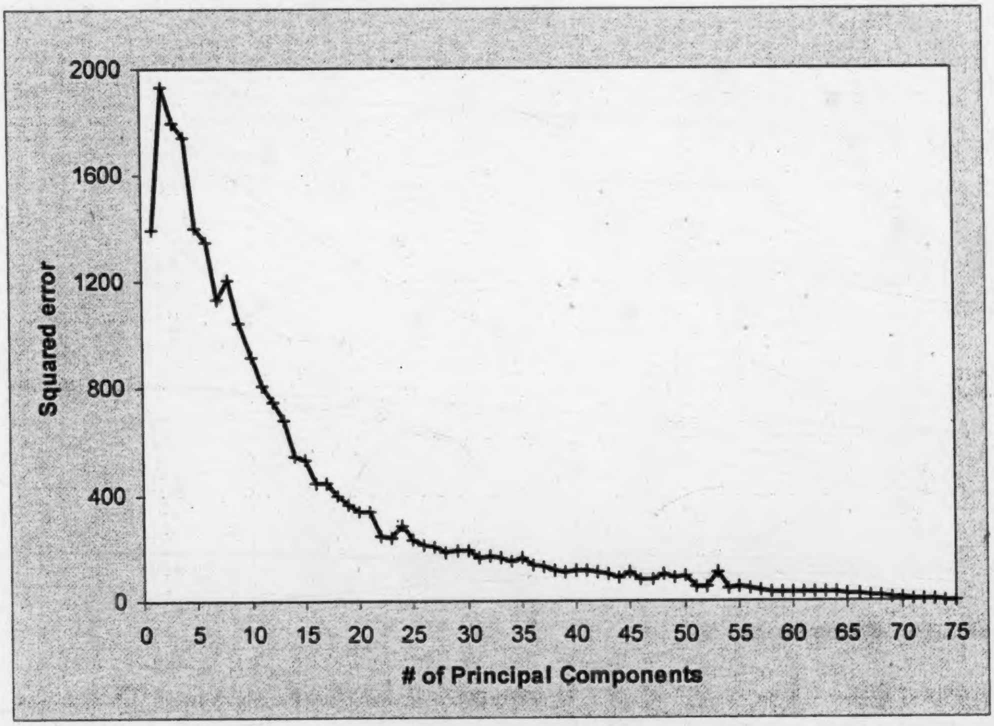


Figure 4-4: Squared Error when using a different number of principal components to approximate the original data set

For less than 25 principal components, the squared error decreased almost linearly with a strong slope. When more than 25 principal components were used, it still decreased almost linearly but with a smaller slope.

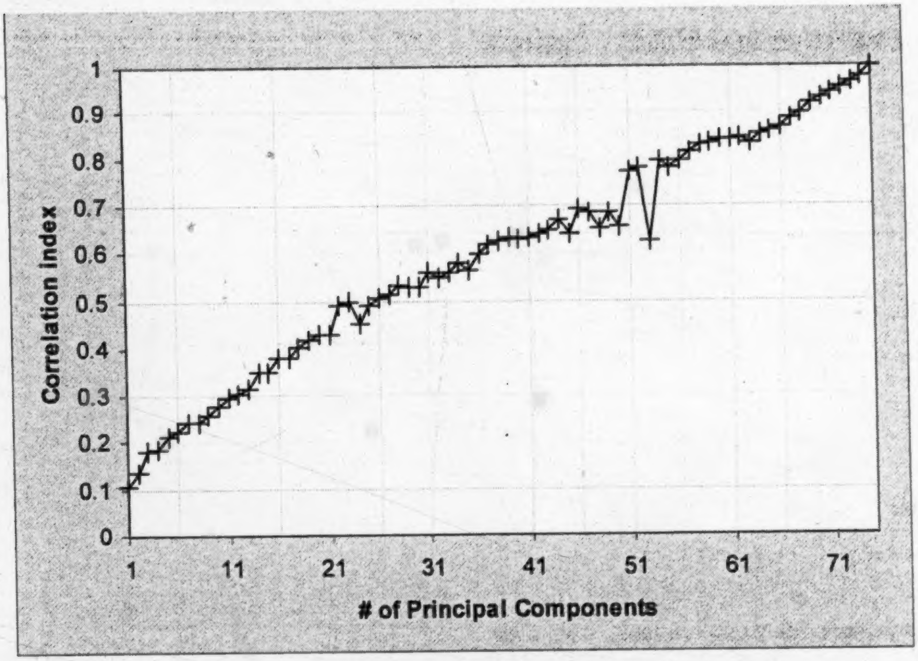


Figure 4-5: Correlation index vs. Number of principal components, averaged for all classes

The correlation index increased linearly with an almost constant slope when the number of principal components increased. There was no indication of a number of principal components where the correlation index would maintain its value.

C. Accuracy when using LDA

Linear discriminant analysis (explained in Section 3.2) is used to classify the data set represented by its principal components. The accuracy of this classification is computed for different quantities of principal components. The results are shown in Figure 4-6.

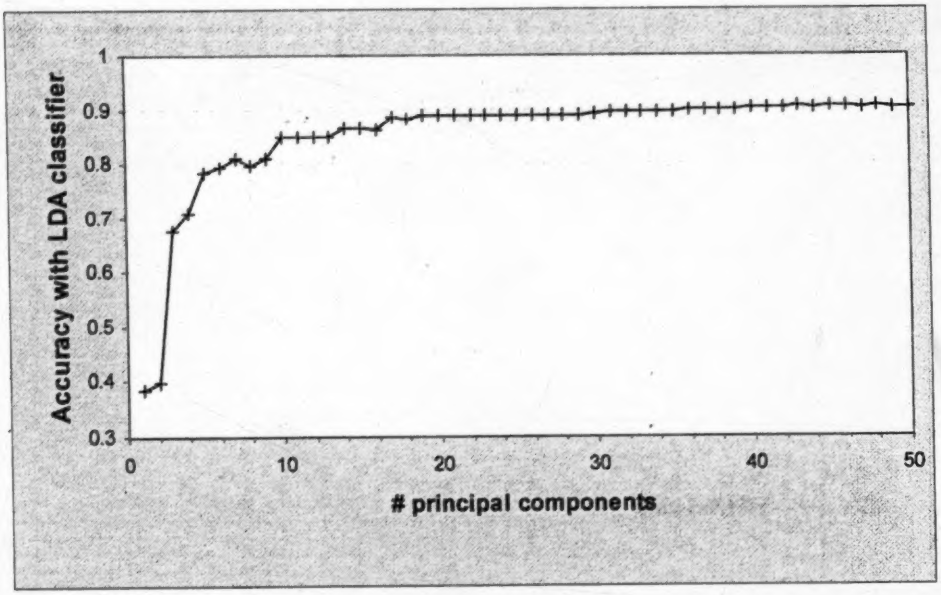


Figure 4-6: Accuracy with the LDA classifier when using different number of principal components for the PCA

The number of principal components affects the accuracy of the classification. When more principal components are used, the accuracy improves. After twenty or more principal components, the accuracy seems to be less affected by the number of principal components.

D. Summary and conclusion

From the different measures of fit presented above, it can be said that the variance of the data is maintained with more than 15 principal components, the squared error of the approximation stabilizes with more than 25 principal components, the correlation index does not give enough information to find a cut-off in the number of principal components, and the accuracy using linear discriminant analysis to classify the data does not improve much for more than 20 principal components.

The number of principal components to use is between 15 and 20 (i.e. the first 15 to 20 principal components.

4.2.2. Feature extraction and selection using Hermite Transformation

The Hermite transformation was used in this work to reduce the number of variables of the original data set formed by QRS segments, by converting the dataset formed by O-dimensional vectors to a dataset formed by n-dimensional vectors as explained in Section

2.5.2.B. As it was shown in Section 2.6.3, the original data set has O= 75 features (73 samples plus two values of local rhythm).

The purpose of this experiment was to estimate the number of Hermite coefficients to use to represent appropriately the original data in this study.

Using the data set described in Section 2.6.4, the experiment consisted in performing the Hermite transformation explained in Section 2.5.2.B to obtain the Hermite coefficients. The selection of the number of coefficients to use for later experiments was based on three series of calculations:

- correlation between the original data set and its approximation using the inverse Hermite transformation with different number of Hermite coefficients.
- squared Error of the approximation with respect to the original data set.
- accuracy when using an LDA classifier for different number of Hermite coefficients.

A. Squared error and correlation index

The Hermite transformation was performed on the original data set using n Hermite coefficients for n={4, 5, 6, 7, 8, 9, 10, 11, 12}. The inverse transformation was performed to obtain an approximation of the original data set for each value of n.

The squared error (explained in Section 2.4.5.A) and the correlation index (explained in Section 2.4.5.B) between both versions were calculated for each approximation. The results are shown in Figure 4-7 and Figure 4-8 respectively.

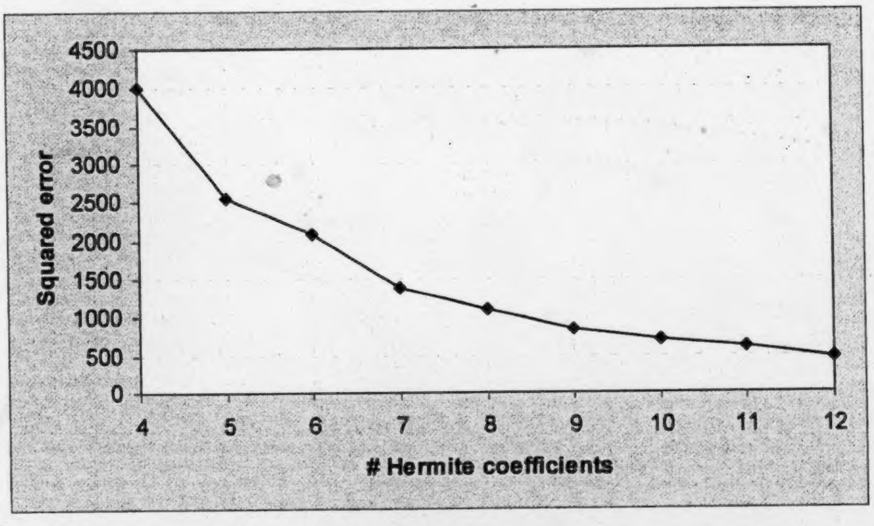


Figure 4-7: Number of Hermite coefficients vs. Squared error between the modeled and the original signals

The squared error decreases when the number of Hermite coefficients increases. The slope seems to be smoother when more than nine coefficients are used.

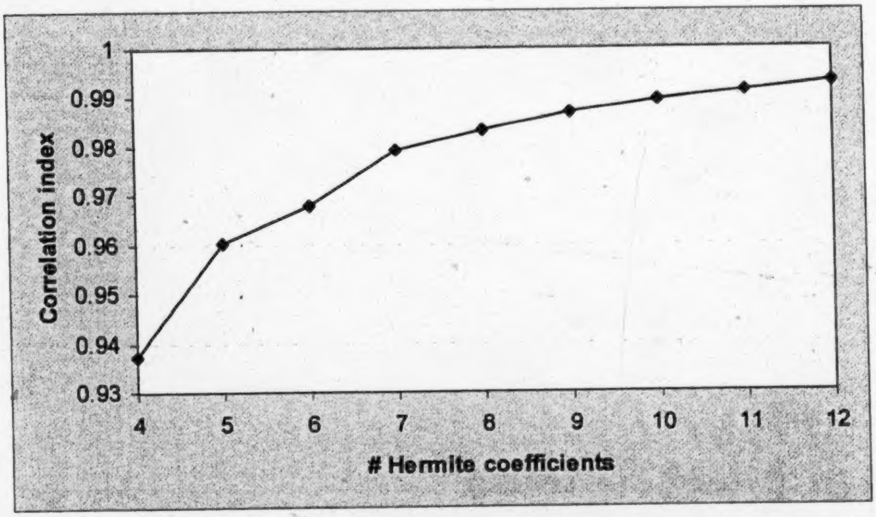


Figure 4-8: Number of Hermite coefficients vs. linear correlation coefficient between the modeled and the original signals

The correlation index increases with the number of Hermite coefficients. It starts to stabilize when ten or more Hermite coefficients are used.

B. Accuracy when using LDA

Linear discriminant analysis (explained in Section 3.2) is used to classify the data set represented by the coefficients obtained from the Hermite transformation. The accuracy of this classification is computed for different quantities of coefficients. The results are shown below in Table 4-1 and Figure 4-9.

When the data set is modeled by 4, 5, 6 Hermite coefficients, the accuracy does not improve. The classification accuracy improves when the number of Hermite coefficients increase from 7 to 10. When 11 and more coefficients are used, the accuracy and the kappa score, with the LDA classifier, decrease.

Table 4-1: Classification accuracy and kappa score using LDA depending on the number of Hermite coefficients

Data set	Number of coefficients	Accuracy	Kappa score
QRS segments	Not Applicable	0.9166	0.8889
Hermite transformation	4	0.7048	0.6064
	5	0.7076	0.6102
	6	0.7020	0.6026
	7	0.7345	0.6459
	8	0.7684	0.6911
	9	0.7726	0.6968
	10	0.7938	0.725
	11	0.7627	0.6836
	12	0.7571	0.6761

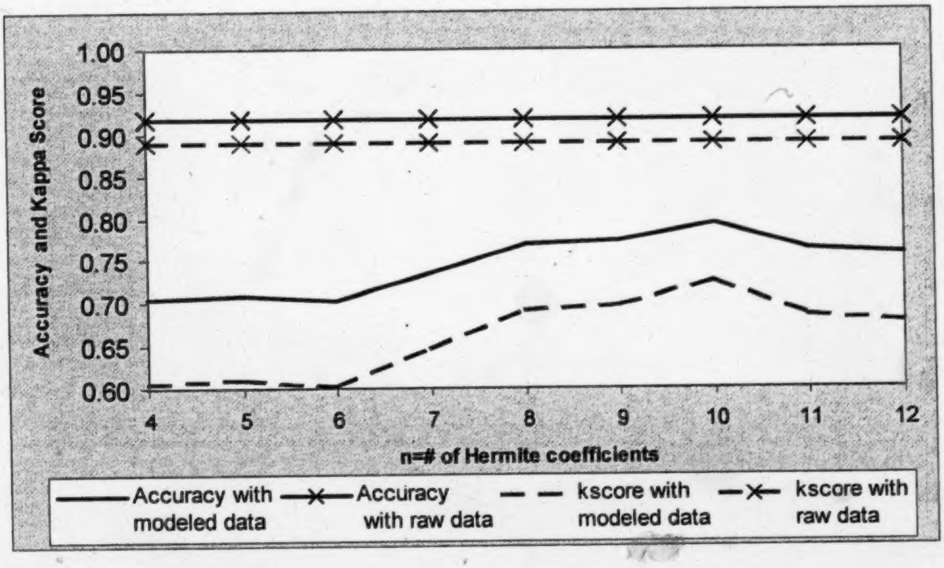


Figure 4-9: Number of Hermite coefficients vs. Accuracy and Kappa score using LDA

C. Summary and conclusion

From the different measures of fit presented above, it can be said that the squared error of the approximation stabilizes with more than 9 Hermite coefficients, the correlation index has a cut-off zone around 10 Hermite coefficients, and the accuracy using linear discriminant analysis to classify the data reaches a peak when using 10 Hermite coefficients. Therefore, the number of Hermite coefficients to use is 10.

4.3. QRS detection

In order to perform the detection of the QRS complexes in the ECG signals, three algorithms were implemented based on the theory presented in Section 2.5.3. These algorithms are the Okada (explained in Section 2.5.3.B), the MOBD (explained in Section 2.5.3.D), and the Neural-networks approach (explained in Section 2.5.3.E). An improved version of the Okada algorithm that combine the original Okada algorithm and the MOBD was implemented.

The experiments were performed on the data set described in Section 2.6.2, using the first minute of all the records to configure the detector and nine minutes to test its performance. The results are shown in the form of:

1. number of true positives (TP): Number of beats that were correctly detected.
2. number of false positives (FP): Number of beats that were incorrectly detected.
3. number of false negatives (FN): Number of beats that were missed.
4. error rate (FP+FN)/(#beats)
5. positive predictivity TP/(TP+FP)

4.3.1. Okada QRS detector

A. Architecture and Methodology

Two sets of experiments were performed using two versions of the Okada algorithm [55]:

1. the algorithm implemented in its original form (as described in Section 2.5.3.B), with its filtering steps and then the basic thresholding to search for the peak of the R wave.
2. the algorithm implemented with the original Okada's filtering step but with an improved thresholding method based on the work by Suppappola and Sun [77],

A preliminary step was performed to obtain the heuristic parameters of the filter, searching values by trial and error to obtain the minimal error rate on the training set. The same parameters were finally used on the two sets of experiments:

- weights of the weighted moving average: [1 2 1];
- parameter of the low pass filter m1b=12 samples (at 360samples/sec);
- parameter of the higher frequency enhancement m=4 samples; and
- range of search for symmetric peaks m= (+/-) 4 samples

In the experiment using the original algorithm, the range to obtain the threshold in the last step is m= (+/-) 4 samples.

The results are presented for the first ten minutes of each record.

B. Results and observations

The results of the experiments with the original algorithm for each record are presented in Table 4-2.

Overall, the algorithm with this particular configuration tends to miss the detection of beats more than detecting extra beats: the number of false negatives (FN) is much higher than of false positives (FP). This translates in a good positive predictivity vs. a poor error rate.

The range of the error rate can vary from very small values, as for record 100 and others, to very high values as in record 108 among others. Record 108 is particularly challenging, because of the poor quality of the signal.

Table 4-2: Results in testing the original Okada QRS detector

RECORD	# of Beats	TP	FP	FN	Error rate (FP+FN) /# beats	Positive predictivity TP / #detected beats
ALL	**36683**	**29381**	**408**	**6574**	**0.1903**	**0.9863**
100	760	760	0	0	0	1
101	645	628	0	17	0.0264	1
102	728					
103	703	703	0	0	0	1
104	743	246	28	497	0.7066	0.8978
105	832	827	0	5	0.0060	1
106	646	370	0	276	0.4272	1
107	706	703	0	3	0.0042	1
108	566	21	61	545	1.0707	0.2561
109	857	669	0	188	0.2194	1
111	697	524	0	173	0.2482	1
112	853	853	0	0	0	1
113	581	581	0	0	0	1
114	556	36	3	520	0.9406	0.9231
115	635	635	0	0	0	1
116	797	767	12	30	0.0527	0.9846
117	504	504	0	0	0	1
118	770	730	30	40	0.0909	0.9605
119	659	527	0	132	0.2003	1
121	608	608	0	0	0	1
122	837	836	0	1	0.0012	1
123	506	505	0	1	0.0020	1
124	524	484	0	40	0.0763	1
200	868	822	20	46	0.0760	0.9762
201	777	744	0	33	0.0425	1
202	534	527	0	7	0.0131	1
203	997	86	8	911	0.9218	0.9149
205	924	894	0	30	0.0325	1
207	794	557	4	237	0.3035	0.9929

RECORD	# of Beats	TP	FP	FN	Error rate (FP+FN) /# beats	Positive predictivity TP / #detected beats
208	1012	557	60	455	0.5089	0.9028
209	1023	1010	4	13	0.0166	0.9961
210	881	823	1	58	0.0670	0.9988
212	932	920	0	12	0.0129	1
213	1098	1009	6	89	0.0865	0.9941
214	762	640	12	122	0.1759	0.9816
215	1131	847	79	284	0.3210	0.9147
217	727	588	7	139	0.2008	0.9882
219	758	725	0	33	0.0435	1
220	698	697	0	1	0.0014	1
221	827	660	2	167	0.2044	0.9970
222	739	518	3	221	0.3031	0.9942
223	839	623	4	216	0.2622	0.9936
228	697	127	2	570	0.8207	0.9845
230	729	696	11	33	0.0604	0.9844
231	677	488	0	189	0.2792	1
232	602	395	50	207	0.4269	0.8876
233	1023	999	1	24	0.0244	0.9990

The results of the improved version are shown in Table 4-3. With this improved version, the results are improved drastically. Even record 108 gives acceptable results, considering its poor quality of signal. This shows that the threshold step is critical for obtaining good results with the algorithm.

Table 4-3: Results using the improved version of the Okada QRS detector

RECORD	# of Beats	TP	FP	FN	Error rate (FP+FN) /# beats	Positive predictivity TP / #detected beats
ALL	**36683**	**34265**	**557**	**1690**	**0.0613**	**0.9840**
100	760	760	0	0	0	1
101	645	641	1	4	0.0078	0.9984
102	728					
103	703	703	0	0	0	1
104	743	738	29	5	0.0458	0.9622
105	832	828	2	4	0.0072	0.9976
106	646	591	0	55	0.0851	1
107	706	706	0	0	0	1
108	566	529	131	37	0.2968	0.8015
109	857	854	1	3	0.0047	0.9988
111	697	697	0	0	0	1
112	853	853	0	0	0	1

RECORD	# of Beats	TP	FP	FN	Error rate (FP+FN) /# beats	Positive predictivity TP / #detected beats
113	581	581	0	0	0	1
114	556	551	5	5	0.0180	0.9910
115	635	635	0	0	0	1
116	797	764	6	33	0.0489	0.9922
117	504	504	0	0	0	1
118	770	752	15	18	0.0429	0.9804
119	659	528	2	131	0.2018	0.9962
121	608	608	1	0	0.0016	0.9984
122	837	836	0	1	0.0012	1
123	506	505	0	1	0.0020	1
124	524	520	3	4	0.0134	0.9943
200	868	860	8	8	0.0184	0.9908
201	777	749	1	28	0.0373	0.9987
202	534	527	0	7	0.0131	1
203	997	881	18	116	0.1344	0.9800
205	924	917	0	7	0.0076	1
207	794	686	81	108	0.2380	0.8944
208	1012	579	125	433	0.5514	0.8224
209	1023	1019	1	4	0.0049	0.9990
210	881	836	0	45	0.0511	1
212	932	932	0	0	0	1
213	1098	1017	8	81	0.0811	0.9922
214	762	735	4	27	0.0407	0.9946
215	1131	1120	8	11	0.0168	0.9929
217	727	720	5	7	0.0165	0.9931
219	758	729	1	29	0.0396	0.9986
220	698	698	0	0	0	1
221	827	670	1	157	0.1911	0.9985
222	739	739	0	0	0	1
223	839	746	83	93	0.2098	0.8999
228	697	672	6	25	0.0445	0.9912
230	729	729	0	0	0	1
231	677	495	0	182	0.2688	1
232	602	601	10	1	0.0183	0.9836
233	1023	1004	0	19	0.0186	1

As a reference, the positive predictivity is shown in Table 4-4 for each class of beat. This table not only emphasizes how better the improved version is with respect to the original, but it also shows how the class of the QRS also influences the ability of the detector to find them. With the original algorithm, the classes are a determining factor for its success. In the improved version, because the threshold adapts to the beat structure, the classes of the QRS are not so relevant.

Table 4-4: Positive predictivity of the Okada QRS detector per class

CLASS	# beats	Original OKADA QRS detector	Improved OKADA QRS detector
V	2168	0.6713	0.8434
R	2419	0.9244	0.9864
L	2623	0.8052	0.9854
N	24263	0.8397	0.9627
A	698	0.7231	0.9806
a	698	0.8077	0.8846
F	628	0.7063	0.7552
f	633	0.3899	0.9856
J	39	0.8387	0.9032
j	39	1	0.7273
P	2158	0.7988	0.9927
Q	182	0.0374	0.1337
!	127	0.1364	0.4773

In Figure 4-10, the error rate is shown for the two versions of the algorithm and for each record. The records were sorted with respect to the results of the improved Okada algorithm. In general, with this change in the thresholding process of the algorithm, the Okada algorithm passes from being a comprehensive but not so accurate algorithm to be a comprehensive AND accurate algorithm for the data used in this study.

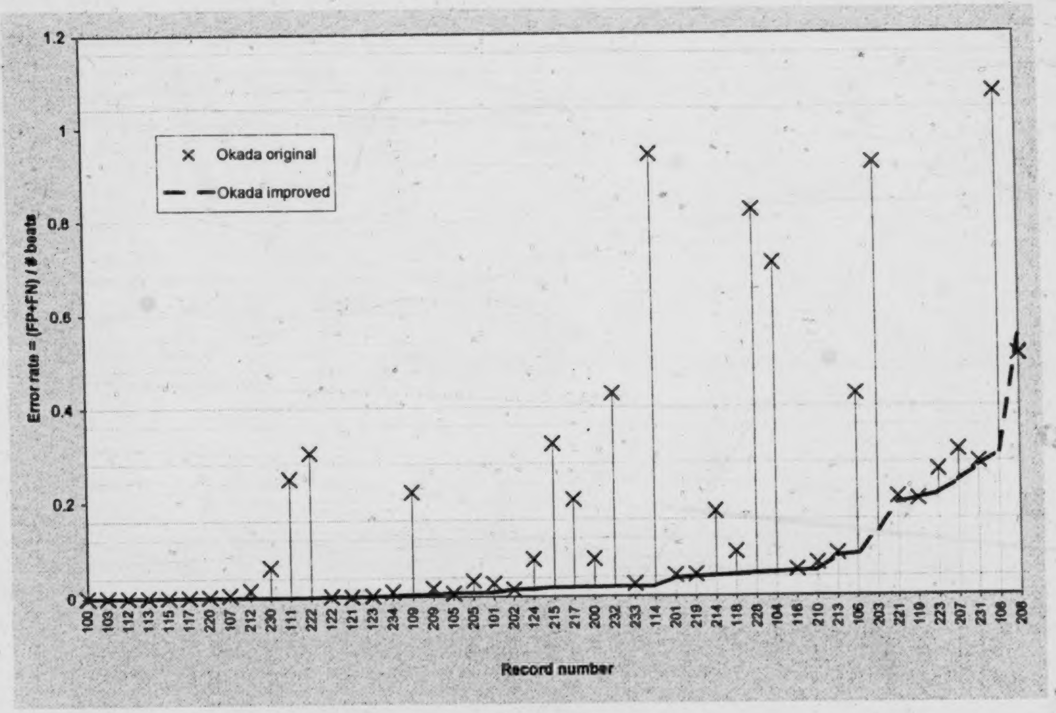


Figure 4-10: QRS detection error rate for each record using the original Okada algorithm and its improved version

4.3.2. MOBD QRS detector

A. Architecture and Methodology

The MOBD algorithm was implemented following the description in Section 2.5.3.D.

The order of the MOBD transformation was chosen to be 4.

The threshold is updated every 36 samples (at 360 samples per sec) or when a QRS is detected.

The experiments were performed using the data set described in Section 2.6.2.

B. Results and observations

The results for each record are shown in Table 4-5.

Overall, the results are less than satisfactory. The number of false positive and false negatives is high and the proportion of false positive and false negatives are comparable. This algorithm under the present configuration seems to detect the QRS outside of the acceptable range.

Table 4-5: Results using the MOBD QRS detector

record	# of Beats	TP	FP	FN	Error rate (FP+FN) /# beats	Positive predictivity TP / #detected beats
ALL	**36683**	**34536**	**1763**	**2147**	**0.1066**	**0.9514**
100	760	760	0	0	0	1
101	645	644	4	1	0.0078	0.9938
102	728	710	18	18	0.0495	0.9753
103	703	703	0	0	0	1
104	743	647	144	96	0.3230	0.8180
105	832	832	2	0	0.0024	0.9976
106	646	632	1	14	0.0232	0.9984
107	706	704	1	2	0.0042	0.9986
108	566	500	257	66	0.5707	0.6605
109	857	855	0	2	0.0023	1
111	697	317	380	380	1.0904	0.4548
112	853	853	0	0	0	1
113	581	581	0	0	0	1
114	556	556	0	0	0	1
115	635	635	0	0	0	1
116	797	797	1	0	0.0013	0.9987
117	504	504	0	0	0	1
118	770	767	0	3	0.0039	1
119	659	571	88	88	0.2671	0.8665
121	608	607	1	1	0.0033	0.9984
122	837	837	0	0	0	1
123	506	505	0	1	0.0020	1
124	524	221	303	303	1.1565	0.4218
200	868	867	1	1	0.0023	0.9988
201	777	744	1	33	0.0438	0.9987
202	534	534	0	0	0	1
203	997	910	26	87	0.1133	0.9722
205	924	919	0	5	0.0054	1
207	794	670	86	124	0.2645	0.8862
208	1012	655	211	357	0.5613	0.7564
209	1023	1019	0	4	0.0039	1
210	881	853	2	28	0.0341	0.9977
212	932	930	0	2	0.0021	1
213	1098	1086	10	12	0.0200	0.9909
214	762	751	4	11	0.0197	0.9947
215	1131	1126	2	5	0.0062	0.9982

record	# of Beats	TP	FP	FN	Error rate (FP+FN) /# beats	Positive predictivity TP / #detected beats
217	727	723	5	4	0.0124	0.9931
219	758	755	0	3	0.0040	1
220	698	698	0	0	0	1
221	827	669	9	158	0.2019	0.9867
222	739	733	3	6	0.0122	0.9959
223	839	830	3	9	0.0143	0.9964
228	697	561	166	136	0.4333	0.7717
230	729	729	0	0	0	1
231	677	505	0	172	0.2541	1
232	602	601	33	1	0.0565	0.9479
233	1023	1009	1	14	0.0147	0.9990
234	921	921	0	0	0	1

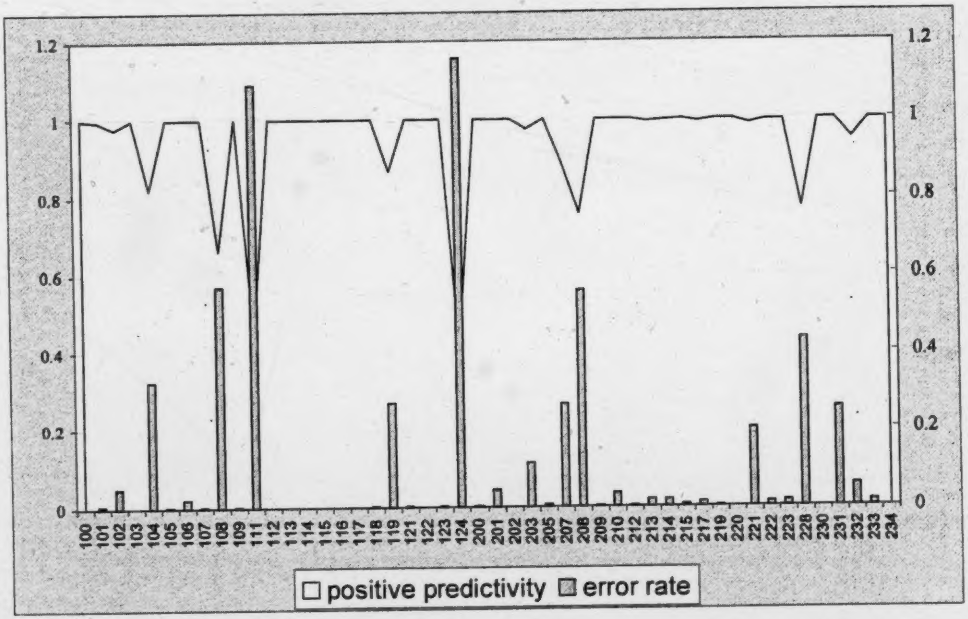


Figure 4-11: Positive predictivity and error rate for the MOBD algorithm

The positive predictivity is, in general, high. In the case of the error rate, there is room for improvement. It is interesting to note that for many of the records the results are very good. The distribution of classes per record in Table 4-6 shows that the error rate is somewhat correlated to the distribution of classes in the record. When the error rate is minimal, the dominant class is normal and not many QRS belong to other classes. There are probably other factors affecting the performance of the MOBD in the records, as for example, the quality of the signal and the shape of the beats. But in general, this algorithm does not give satisfactory results on this data set.

Table 4-6: Distribution of classes of records sorted by error rate with the MOBD algorithm

ErrorRate	record	N	L	R	A	a	J	S	V	F	!	e	j	E	P	f	Q
0	100	750	0	0	5	5	0	0	0	0	0	0	0	0	0	0	0
0	103	690	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	112	827	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	113	559	0	0	3	3	0	0	0	0	0	0	0	0	0	0	0
0	114	502	0	0	0	0	0	0	32	2	0	0	0	0	0	2	0
0	115	615	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	117	489	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	122	811	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	202	510	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0
0	220	654	0	0	23	23	0	0	0	0	0	0	0	0	0	0	0
0	230	706	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	234	892	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.001255	116	733	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0
0.001976	123	489	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.002146	212	241	0	661	0	0	0	0	0	0	0	0	0	0	0	0	0
0.002304	200	601	0	0	4	4	0	0	233	0	0	0	0	0	0	0	0
0.002334	109	0	836	0	0	0	0	0	11	1	0	0	0	0	0	2	0
0.002404	105	787	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0
0.003289	121	590	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.003896	118	0	0	708	27	27	0	0	7	0	0	0	0	0	0	0	0
0.00391	209	861	0	0	129	129	0	0	0	0	0	0	0	0	0	0	0
0.003958	219	703	0	0	3	3	0	0	25	0	0	0	0	0	0	0	1
0.004249	107	0	0	0	0	0	0	0	4	0	0	0	0	0	678	0	0
0.005411	205	872	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0
0.006189	215	935	0	0	0	0	0	0	55	0	0	0	0	0	0	0	0
0.007752	101	623	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0.012179	222	702	0	0	7	7	5	0	0	0	0	0	5	0	0	0	0
0.01238	217	20	0	0	0	0	0	0	53	68	0	0	0	0	549	68	0
0.014303	223	716	0	0	27	27	0	0	58	6	0	4	0	4	0	6	0
0.014663	233	730	0	0	2	2	0	0	259	6	0	0	0	0	0	6	0
0.019685	214	0	651	0	0	0	0	0	85	0	0	0	0	0	0	0	1
0.020036	213	775	0	0	9	9	0	0	64	141	0	0	0	0	0	141	0
0.02322	106	566	0	0	0	0	0	0	62	0	0	0	0	0	0	0	0
0.034052	210	797	0	0	2	2	0	0	49	2	0	0	0	0	0	2	0
0.043758	201	692	0	0	19	19	6	0	19	0	0	0	6	0	0	0	0
0.049451	102	95	0	0	0	0	0	0	2	29	0	0	0	0	581	29	0
0.056478	232	0	0	144	436	436	0	0	0	0	0	0	0	0	0	0	0
0.11334	203	811	0	0	1	1	0	0	152	0	0	0	0	0	0	0	0
0.201935	221	646	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0
0.254062	231	115	0	369	0	0	0	0	1	0	0	0	0	0	0	0	163

ErrorRate	record	N	L	R	A	a	J	S	V	F	!	e	j	E	P	f	Q
0.264484	207	0	**451**	**82**	0	0	0	0	**95**	0	**127**	0	0	0	0	0	0
0.267071	119	**503**	0	0	0	0	0	0	**134**	0	0	0	0	0	0	0	0
0.323015	104	**100**	0	0	0	0	0	0	**1**	**242**	0	0	0	0	**350**	**245**	**17**
0.433286	228	**524**	0	0	0	0	0	0	**151**	0	0	0	0	0	0	0	0
0.561265	208	**494**	0	0	0	0	0	0	**351**	**130**	0	0	0	0	0	**130**	0
0.570671	108	**537**	0	0	0	0	0	0	**6**	0	0	0	0	0	0	**1**	0
1.090387	111	0	**685**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1.156489	124	0	0	**455**	0	0	**28**	0	**19**	**1**	0	0	**28**	0	0	**1**	0

4.3.3. Neural networks QRS detector

A. Architecture and Methodology

The algorithm explained in Section 2.5.3.E was implemented with the following parameters:

- Number of inputs of the neural networks: 73
- Number of neurons in the hidden layer: 36
- Scaling factor for the sigmoid: 5

The training process was performed with the following characteristics:

- Maximum number of epochs: 50
- Learning rate was decreased proportionally after each epoch.
- Initial learning rate: 0.2
- Final learning rate: 0.00005
- Training segments: Sliding window with 54 samples overlap + all the windows where the R peak was in the center of the window (all the positive cases were taken into account).
- There were two options for the output of the training set:
    1. 1 when the R peak was in the center of the window, 0 otherwise.
    2. Quadratic equation depending on the position of the peak of the R wave with respect to the center of the window (reaching 1 when it is in the center and 0 when it is 36 samples from the center and beyond)

Because the search for the weights is a gradient descent search, the algorithm can find local minima. This is why the experiments were performed five times with different randomly selected initial weights and the configuration with the best results in training was chosen. The results in training are for this configuration.

B. Results and observations

The results are shown in Table 4-7 for the network trained with the first output option, and Table 4-8 for the network trained with the second output option.

Overall, the results look promising. The number of false positives is very low except for records 124, 207, and 108. The number of missed beats is low except for records 200, 207, 223, and 231.

Table 4-7: Results using the Neural networks QRS detector trained with option1

record	# of Beats	TP	FP	FN	Error rate (FP+FN) /# beats	Accuracy in detection TP / #detected beats
ALL	**36683**	**34885**	**848**	**1798**	**0.0721**	**0.9763**
100	760	760	0	0	0	1
101	645	557	7	88	0.1473	0.9876
102	728	686	17	42	0.0810	0.9758
103	703	702	0	1	0.0014	1
104	743	728	68	15	0.1117	0.9077
105	832	825	2	7	0.0108	0.9976
106	646	600	30	46	0.1176	0.9479
107	706	703	3	3	0.0085	0.9958
108	566	521	85	45	0.2297	0.8527
109	857	856	1	1	0.0023	0.9988
111	697	692	1	5	0.0086	0.9986
112	853	846	22	7	0.0340	0.9747
113	581	557	0	24	0.0413	1
114	556	548	11	8	0.0342	0.9803
115	635	630	6	5	0.0173	0.9906
116	797	796	1	1	0.0025	0.9987
117	504	503	1	1	0.0040	0.9980
118	770	761	24	9	0.0429	0.9694
119	659	657	2	2	0.0061	0.9970
121	608	608	9	0	0.0148	0.9854
122	837	834	0	3	0.0036	1
123	506	503	1	3	0.0079	0.9980
124	524	466	112	58	0.3244	0.7832
200	868	660	1	208	0.2408	0.9985
201	777	724	10	53	0.0811	0.9864
202	534	534	0	0	0	1
203	997	916	52	81	0.1334	0.9463
205	924	897	22	27	0.0530	0.9761
207	794	666	124	128	0.3174	0.8335

record	# of Beats	TP	FP	FN	Error rate (FP+FN) /# beats	Accuracy in detection TP / #detected beats
208	1012	1006	31	6	0.0366	0.9701
209	1023	971	10	52	0.0606	0.9898
210	881	822	12	59	0.0806	0.9856
212	932	900	0	32	0.0343	1
213	1098	1087	70	11	0.0738	0.9379
214	762	701	4	61	0.0853	0.9943
215	1131	1116	1	15	0.0141	0.9991
217	727	708	6	19	0.0344	0.9916
219	758	694	35	64	0.1306	0.9520
220	698	698	0	0	0	1
221	827	818	2	9	0.0133	0.9976
222	739	738	0	1	0.0014	1
223	839	662	28	177	0.2443	0.9594
228	697	610	4	87	0.1306	0.9935
230	729	711	0	18	0.0247	1
231	677	484	30	193	0.3294	0.9380
232	602	596	0	6	0.0100	1
233	1023	946	0	77	0.0753	1
234	921	881	3	40	0.0467	0.9966

With the second option of outputs in training the results were not improved. On the contrary, the number of false positives increased drastically. The idea behind this option was to “help” the detector detect the beats by anticipating the trend before the peak of the R wave was in the center in the window, and thereby reduce the number of missed beats. This was accomplished, but the number of false negatives was not decreased as much as the number of false positive increased.

Table 4-8: Results using the Neural networks QRS detector trained with option2

record	# of Beats	TP	FP	FN	Error rate (FP+FN) /# beats	Accuracy in detection TP / #detected beats
ALL	**36683**	**35668**	**3532**	**1015**	**0.1240**	**0.9099**
100	760	760	3	0	0.0039	0.9961
101	645	636	23	9	0.0496	0.9651
102	728	713	16	15	0.0426	0.9781
103	703	703	5	0	0.0071	0.9929
104	743	729	158	14	0.2315	0.8219

record	# of Beats	TP	FP	FN	Error rate (FP+FN) /# beats	Accuracy in detection TP / #detected beats
105	832	832	15	0	0.0180	0.9823
106	646	625	5	21	0.0402	0.9921
107	706	704	29	2	0.0439	0.9604
108	566	541	347	25	0.6572	0.6092
109	857	854	1	3	0.0047	0.9988
111	697	697	118	0	0.1693	0.8552
112	853	844	26	9	0.0410	0.9701
113	581	580	73	1	0.1274	0.8882
114	556	556	5	0	0.0090	0.9911
115	635	633	37	2	0.0614	0.9448
116	797	797	2	0	0.0025	0.9975
117	504	500	8	4	0.0238	0.9843
118	770	764	39	6	0.0584	0.9514
119	659	657	2	2	0.0061	0.9970
121	608	608	1	0	0.0016	0.9984
122	837	837	0	0	0	1
123	506	504	1	2	0.0059	0.9980
124	524	488	101	36	0.2615	0.8285
200	868	859	32	9	0.0472	0.9641
201	777	755	35	22	0.0734	0.9557
202	534	527	8	7	0.0281	0.9850
203	997	908	275	89	0.3651	0.7675
205	924	902	26	22	0.0519	0.9720
207	794	595	1113	199	1.6524	0.3484
208	1012	1005	18	7	0.0247	0.9824
209	1023	1021	52	2	0.0528	0.9515
210	881	834	23	47	0.0795	0.9732
212	932	928	1	4	0.0054	0.9989
213	1098	1086	248	12	0.2368	0.8141
214	762	753	23	9	0.0420	0.9704
215	1131	1128	113	3	0.1026	0.9089
217	727	721	79	6	0.1169	0.9013
219	758	756	39	2	0.0541	0.9509
220	698	698	4	0	0.0057	0.9943
221	827	825	99	2	0.1221	0.8929
222	739	731	38	8	0.0622	0.9506
223	839	731	21	108	0.1538	0.9721
228	697	681	125	16	0.2023	0.8449

record	# of Beats	TP	FP	FN	Error rate (FP+FN) /# beats	Accuracy in detection TP / #detected beats
230	729	729	12	0	0.0165	0.9838
231	677	505	4	172	0.2600	0.9921
232	602	601	29	1	0.0498	0.9540
233	1023	910	100	113	0.2082	0.9010
234	921	917	0	4	0.0043	1

4.3.4. Summary of QRS detection results and discussion

The results of the QRS detection are summarized in Table 4-9. For better visualization of the same information, Figure 4-12 and Figure 4-13 are presented.

Surprisingly, the simplest algorithm, the Okada algorithm when its thresholding process was optimized, outperformed the others.

This particular finding shows that a good thresholding process must accompany the method of processing the signal in order to give good results. Note that the original Okada algorithm had the worse error rate, and only with the change in the last step of the process (thresholding), borrowed from the MOBD algorithm, highly outperformed the others (including the MOBD algorithm).

The neural networks configuration gave the second best results, close to the results by the improved Okada algorithm. In Figure 4-14 and Figure 4-15, the results of the three algorithms are shown for every record. Note, that the results of the neural networks are better than the improved Okada algorithm for many records. The advantage of the neural network over the other algorithms is its ability to learn from the data, and the heuristics only come with the learning process.

In order to improve the results of the Okada algorithm, a learning step could be included to optimize the parameters of the digital filtering process.

Table 4-9: Summary of results for QRS detection

	TP	FP	FN	Error rate (FP+FN) /# beats	Positive predictivity TP / #detected beats
MOBD	34536	1763	2147	0.1066	0.9514
NN option1	34885	848	1798	0.0721	0.9763
NN option2	35668	3532	1015	0.1240	0.9099
Okada original	29381	408	6574	0.1903	**0.9863**
Okada improved	34265	557	1690	**0.0613**	0.9840

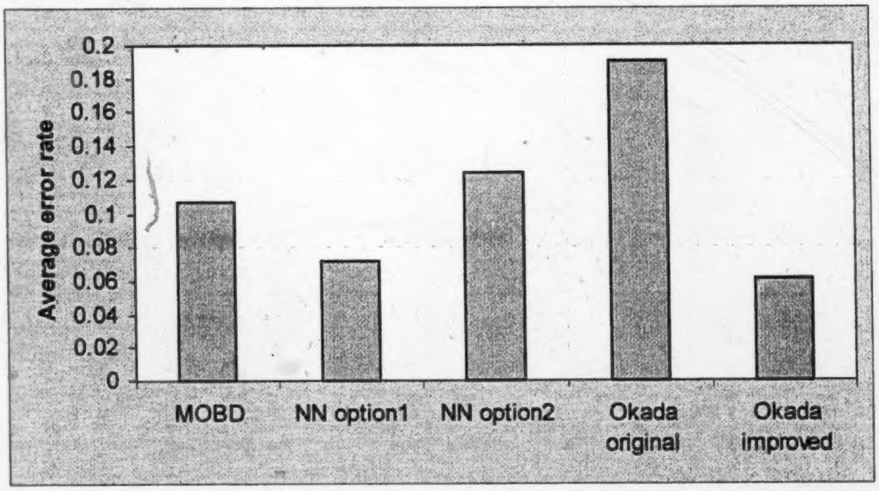


Figure 4-12: Average error rate QRS detection for all the implemented algorithms

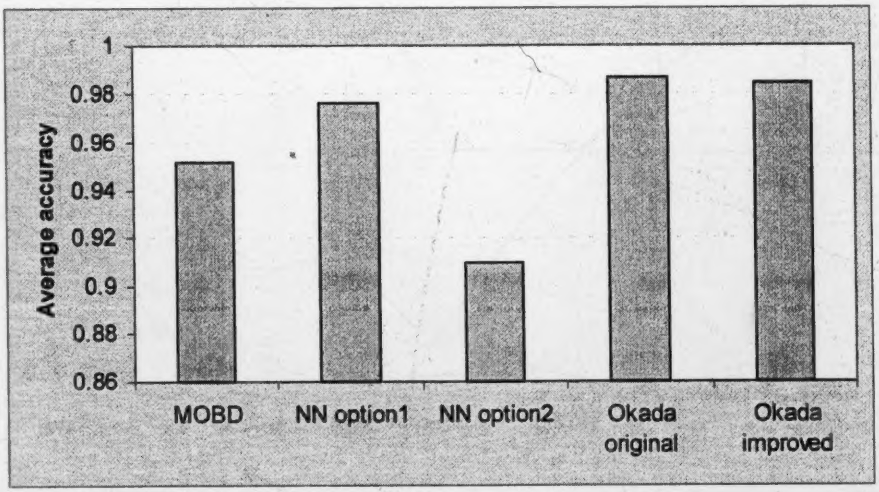


Figure 4-13: Average accuracy QRS detection for all the implemented algorithms

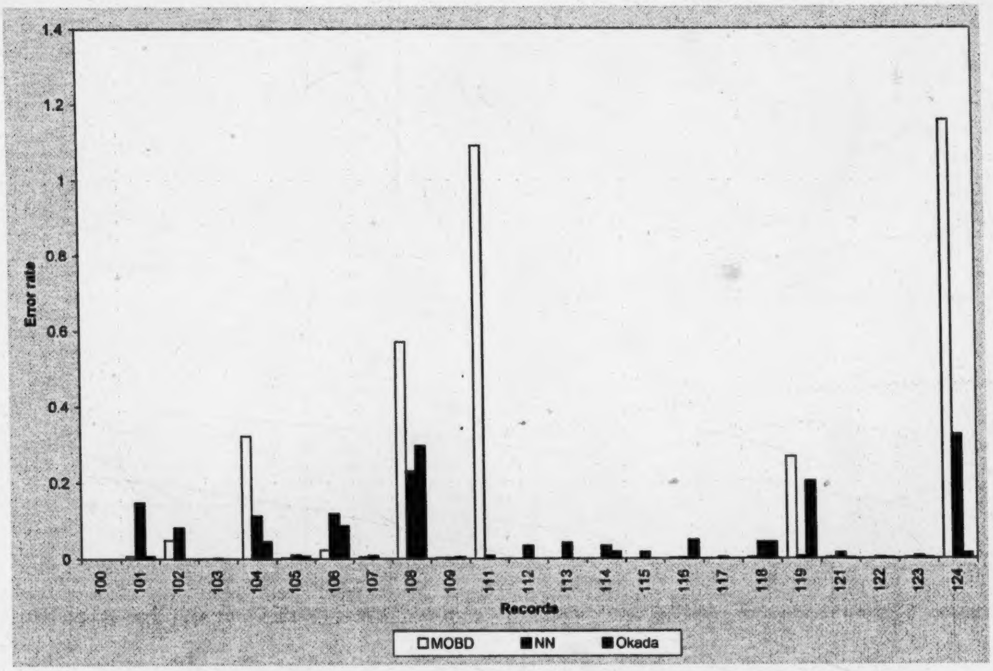


Figure 4-14: Best error rate per record (100-series) for each QRS detection algorithm

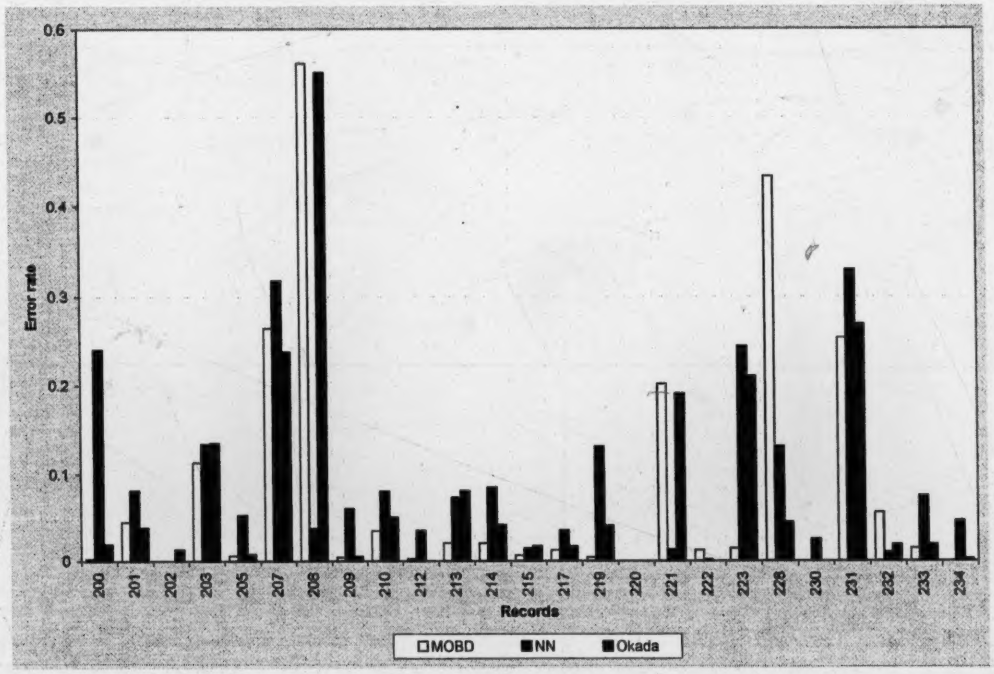


Figure 4-15: Best error rate per record (200-series) for each QRS detection algorithm

4.4. QRS classification

4.4.1. SOM as classifier

A. Architecture and Methodology

Self Organizing Maps (SOM) is a clustering method. In this study it is adapted to work as a classifier. The method and its adaptation to the classification problem is explained in Section 3.3.

The clustering process was performed using the SOM tool presented by Bai in [5].

The configuration of the SOM is the following:

- The measure of distance is Euclidean.
- The neighborhood kernel is Gaussian.
- The size of the neighborhood is initially 3, with a decline rate of 0.1.
- The neurons have hexagonal shape.
- The number of epoch for the rough training is 5000.
- The number of epoch for the fine-tuning of the training is 10000.
- The learning rate is variable:
  - During the rough training, it starts at 1 and decreases to 0.2 throughout the mentioned number of epoch.
  - During the fine-tuning, it starts at 0.2 and ends at 0.01.
- The Initialization of the neurons is random.
- The size of the maps is set to 5 by 5, 10 by 10, 15 by 15, 20 by 20, 25 by 25 and 30 by 30.

The experiments are performed on the data set formed by QRS segments described in Section 2.6.3 and also on its transformed version using 10 Hermite coefficients. The Hermite transformation is explained in Section 2.5.2.B and the number of coefficients to use is determined in Section 4.2.2.

The data set was divided in a training subset of 3184 elements corresponding to 60% of the data set (i.e. 873 class N, 501 class V, 855 class R and 955 class L) and a testing subset with the remaining 40% of the data (2126 elements: 583 class N, 334 class V, 571 class R, and 638 class L).

The normalization of the data was either logistic or none.

In summary, the values that change from one experiment to the other are:

- size of the maps (5 by 5, 10 by 10, 15 by 15, 20 by 20, 25 by 25 and 30 by 30).

- type of data set (original data set with 75 features, or transformed data of 10 Hermite coefficients).
- type of normalization (none or logistic).

B. Results

The results are shown in the form of average entropy per class and of accuracy and kappa score of the classifier for the training set. For the testing set, the accuracy and kappa score of the classifier are presented.

The entropy is calculated for each neuron of the map following the formula presented in Section 2.4.3. The value of entropy obtained is then assigned to the winning class of the corresponding neuron. Finally, the entropy is averaged per class.

The detailed results of the SOM (i.e. cluster map, winning class per neuron, and entropy per neuron for the training set; and confusion matrix, accuracy and kappa score for training and testing sets) are presented for the architecture with best results (minimal entropy and highest accuracy).

Table 4-10: Average entropy per class in the SOM classifier for the Hermite transformed data set with no normalization

Size of MAP	N	V	R	L	All
05x05	1.41	0.97	1.30	1.33	1.25
10x10	0.79	0.59	0.72	0.73	0.71
15x15	0.39	0.36	0.38	0.37	0.37
20x20	0.24	0.33	0.29	0.20	0.27
25x25	0.21	0.31	0.13	0.10	0.19
30x30	0.08	0.25	0.08	0.06	0.12

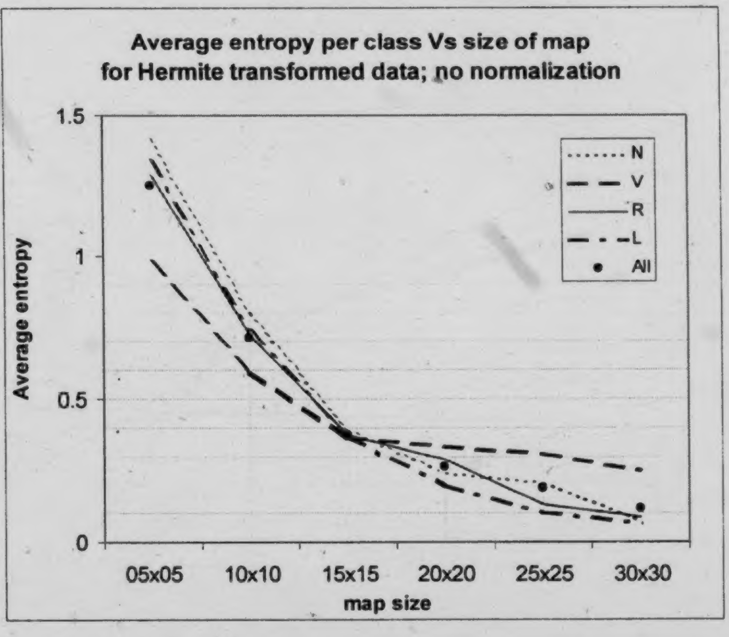


Figure 4-16: Average entropy per class vs. size of SOM map. Data set of Hermite coefficients, with no normalization

Observations on Table 4-10 and Figure 4-16:

- Increasing the size of the map improves the average entropy for all classes.
- For class V, the entropy does not improve much for maps bigger than 15 by 15 neurons. In fact, class V is the class with the worst entropy for maps bigger than 15 by 15 neurons.
- Classes N, R and L maintain relatively close values of average entropy among themselves.

Table 4-11: Average entropy per class in the SOM classifier for the Hermite transformed data set with logistic normalization

Size of MAP	N	V	R	L	All
05x05	1.31	1.18	1.28	0.78	1.14
10x10	0.64	0.36	0.56	0.51	0.52
15x15	0.33	0.23	0.25	0.32	0.28
20x20	0.19	0.15	0.22	0.18	0.18
25x25	0.10	0.14	0.06	0.15	0.11
30x30	0.07	0.11	0.07	0.07	0.08

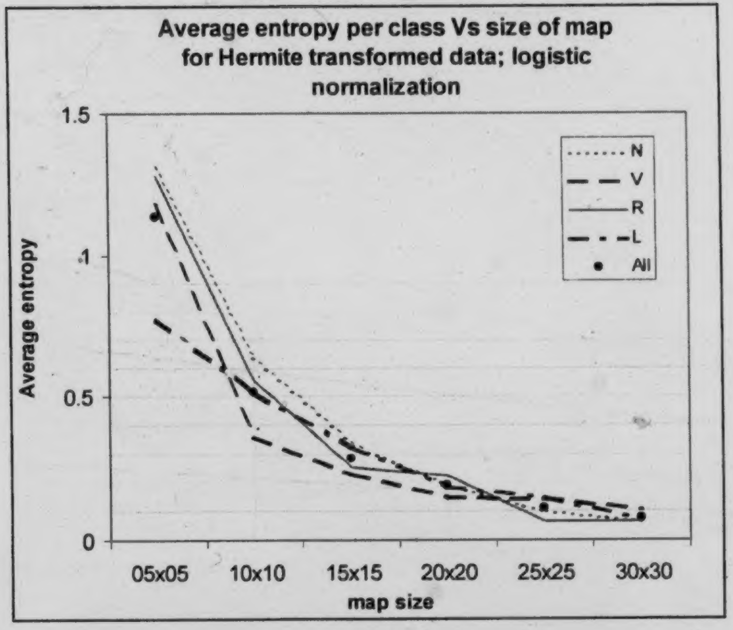


Figure 4-17: Average entropy per class vs. size of SOM map. Data set of Hermite coefficients, with logistic normalization

Observations on Table 4-11 and Figure 4-17:

- Increasing the size of the map improves the average entropy for all classes.
- There is no particular class for which the entropy is better than for the others over all the sizes of maps.

Table 4-12: Average entropy per class in the SOM classifier for the data set formed by QRS segments with no normalization.

Size of MAP	N	V	R	L	All
05x05	0.97	1.20	0.91	1.03	1.03
10x10	0.51	0.39	0.37	0.40	0.42
15x15	0.18	0.34	0.23	0.16	0.23
20x20	0.09	0.30	0.08	0.13	0.15
25x25	0.08	0.22	0.04	0.05	0.10
30x30	0.05	0.17	0.04	0.02	0.07

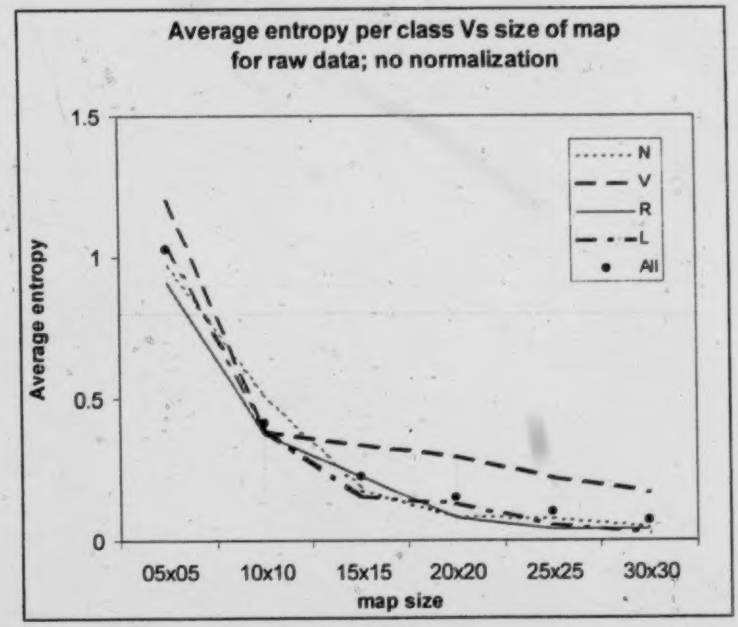


Figure 4-18: Average entropy per class vs. size of SOM map. Data set of QRS segments, with no normalization

Observations on Table 4-11 and Figure 4-18:

- Increasing the size of the map improves the average entropy for all classes.
- The average entropy for class V is generally worse than for the other classes. This difference is accentuated on bigger maps, where the entropy does not improve much when the size of the map is increased.

Table 4-13: Average entropy per class in the SOM classifier for the data set formed by QRS segments with logistic normalization.

Size of MAP	N	V	R	L	All
05x05	0.93	1.05	0.84	0.71	0.88
10x10	0.34	0.69	0.32	0.21	0.39
15x15	0.13	0.43	0.13	0.16	0.21
20x20	0.08	0.22	0.07	0.07	0.11
25x25	0.03	0.15	0.04	0.05	0.07
30x30	0.04	0.09	0.03	0.02	0.05

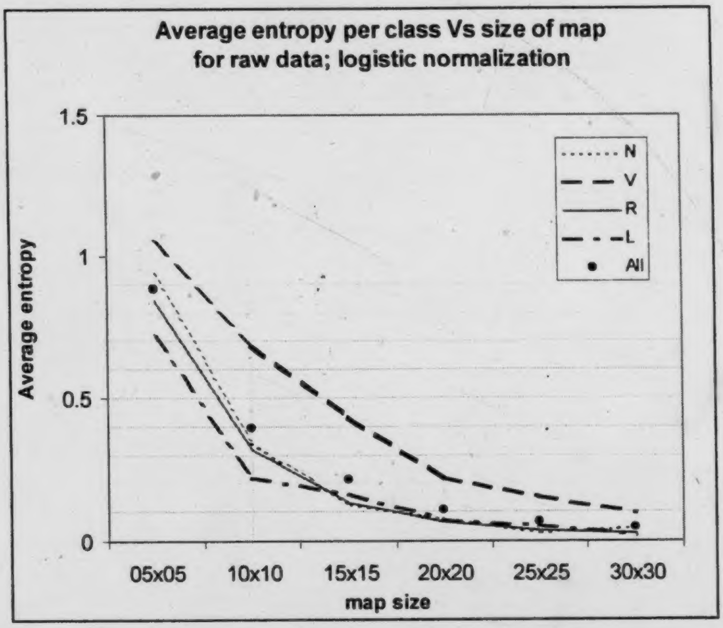


Figure 4-19: Average entropy per class vs. size of SOM map. Data set of QRS segments, with logistic normalization

Observations on Table 4-13 and Figure 4-19:

- Increasing the size of the map improves the average entropy for all classes.
- For class V, the entropy is significantly worse than for other classes.

Table 4-14: Average entropy in the SOM classifier depending on the type of data set.

Size of MAP	Hermite data with no normalization	Hermite data with logistic normalization	QRS segments with no normalization	QRS segments with logistic normalization
05x05	1.25	1.14	1.03	0.88
10x10	0.71	0.52	0.42	0.39
15x15	0.37	0.28	0.23	0.21
20x20	0.27	0.18	0.15	0.11
25x25	0.19	0.11	0.10	0.07
30x30	0.12	0.08	0.07	0.05

These are the observations from Table 4-14:

- The best average entropy is obtained on the data with logistic normalization for all the sizes of SOM.

- The entropy is better for the clusters obtained with the QRS segments than with the data transformed to Hermite coefficients.
- The entropy decreases when the size of the map increases.
- **The best entropy is obtained using the 30 by 30 SOM on the QRS segments with logistic normalization.**

Table 4-15: Accuracy of the SOM classifier in training

	Accuracy in training			
Size Of map	Raw No normalization	Raw Logistic normalization	Hermite No normalization	Hermite Logistic normalization
05x05	0.69	0.76	0.60	0.66
10x10	0.90	0.93	0.83	0.86
15x15	0.95	0.96	0.90	0.92
20x20	0.96	0.98	0.937	0.95
25x25	0.97	0.98	0.945	0.966
30x30	0.98	0.99	0.96	0.975

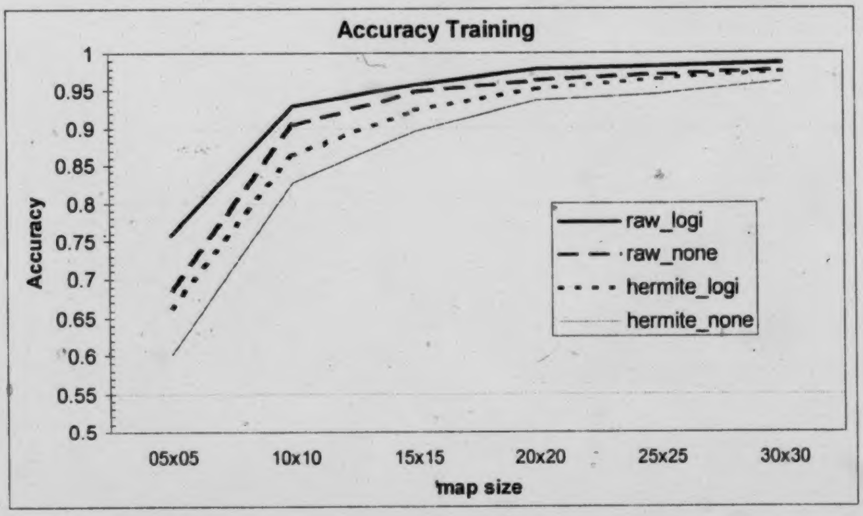


Figure 4-20: Size of the SOM vs. accuracy of the SOM classifier in training

Table 4-16: Accuracy of the SOM classifier in testing.

	Accuracy in testing			
Size Of map	Raw No normalization	Raw Logistic normalization	Hermite No normalization	Hermite Logistic normalization
05x05	0.69	0.78	0.60	0.66
10x10	0.91	0.92	0.83	0.87
15x15	0.93	0.95	0.88	0.92
20x20	0.942	0.96	0.905	0.93
25x25	0.943	0.95	0.903	0.94
30x30	0.938	0.95	0.93	0.937

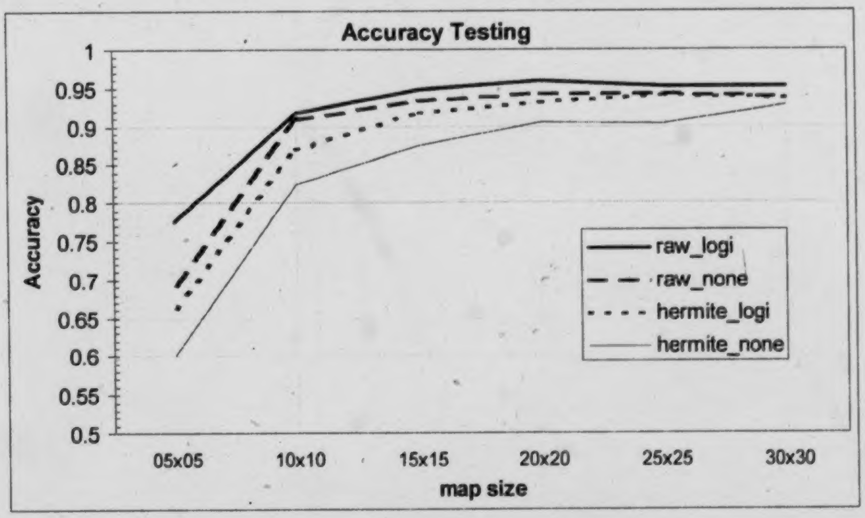


Figure 4-21: Size of the SOM vs. accuracy in testing

The best accuracy is obtained on the data set formed by QRS segments with logistic normalization using the SOM classifier size 30 x 30: 98.65% accuracy in training and 95.16% in testing. This result matches the result with the average entropy of the neurons presented before.

Table 4-17: Kappa score of the SOM classifier in training

	Kappa score in training			
Size Of map	Raw No normalization	Raw Logistic normalization	Hermite No normalization	Hermite Logistic normalization
05x05	0.5745	0.6693	0.4425	0.5430
10x10	0.8693	0.9052	0.7653	0.8152
15x15	0.9310	0.9429	0.8589	0.8961
20x20	0.9515	0.9706	0.9140	0.9356
25x25	0.9622	0.9766	0.9251	0.9540
30x30	0.9698	0.9817	0.9477	0.9655

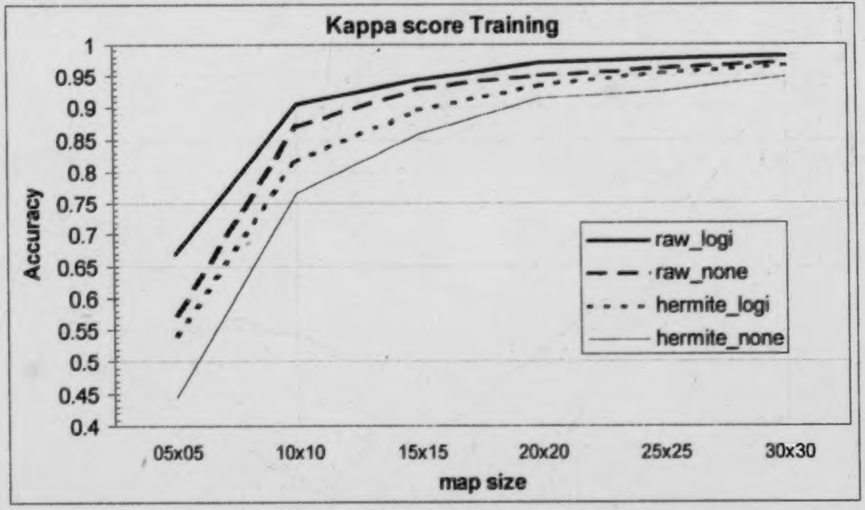


Figure 4-22: Size of the SOM vs. kappa score in training

Table 4-18: Kappa score of the SOM classifier in testing.

	Kappa score in testing			
Size Of map	Raw No normalization	Raw Logistic normalization	Hermite No normalization	Hermite Logistic normalization
05x05	0.5834	0.6930	0.4442	0.5419
10x10	0.8761	0.8892	0.7625	0.8237
15x15	0.9102	0.9280	0.8314	0.8893
20x20	0.9220	0.9442	0.8716	0.9067
25x25	0.9239	0.9348	0.8710	0.9213
30x30	0.9173	0.9358	0.9030	0.9171

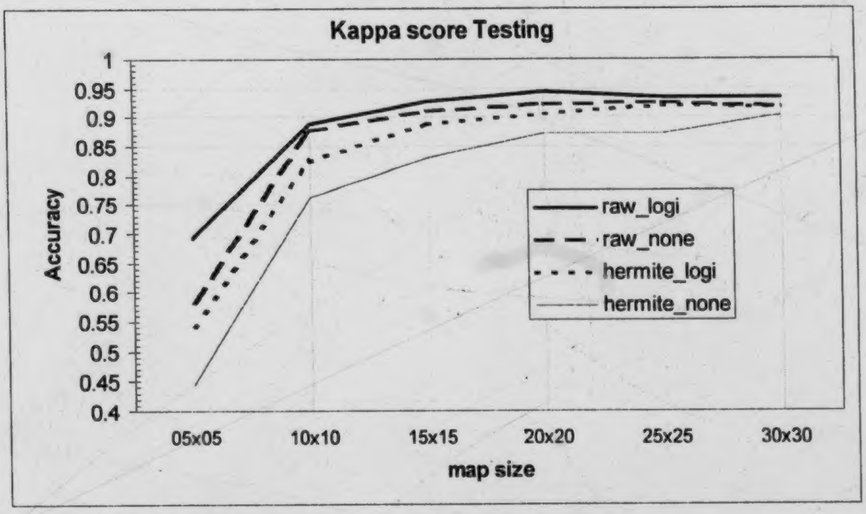


Figure 4-23: Size of the SOM vs. kappa score in testing.

a. Detailed results on the best configuration:

As mentioned above, the best results were obtained with the QRS segments with logistic normalization using the SOM classifier 30 x 30. The complete results of this experiment are shown in Figure 4-24.

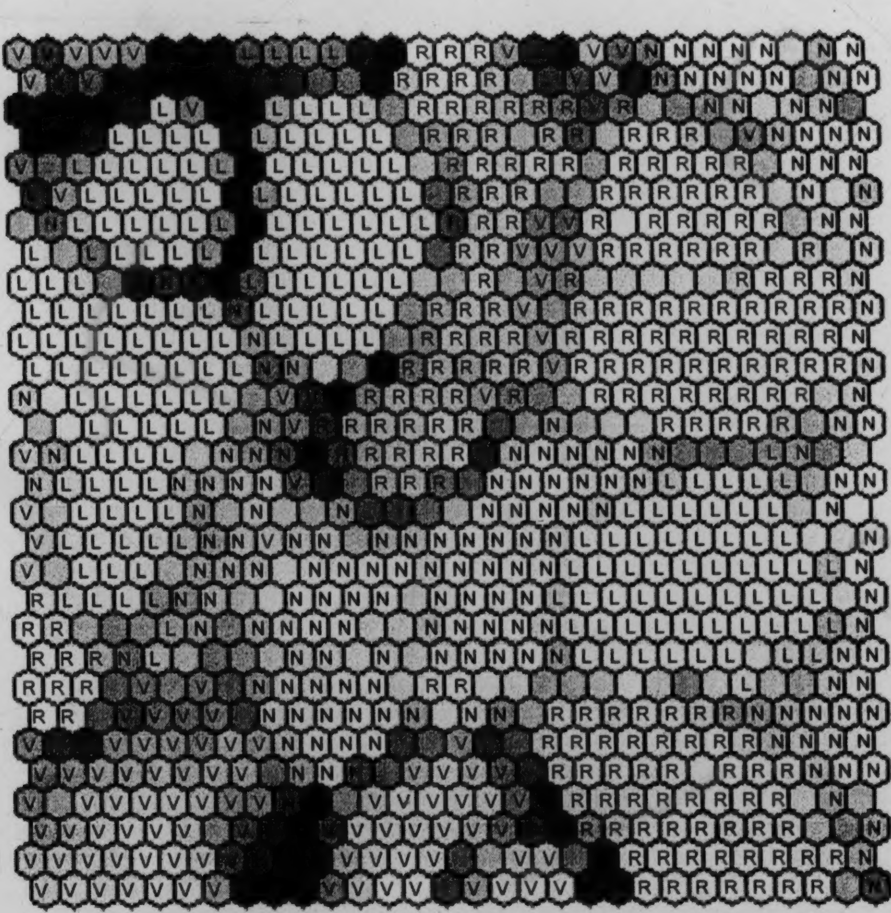


Figure 4-24: SOM classifier 30x30. Cluster map showing the winning class per neuron

Observations on the maps:

As explained in Section 3.3, samples that are similar are mapped to the same neurons. The degree of similarity among the samples is given by the measure of distance (Euclidean in this case) given by the formulas in Table 3-2. The closer the samples are mapped to a neuron, the brighter the color of the neuron. The map shows several bright zones that denote clear groups of clusters.

The brightest groups of clusters are labeled as L. It means that the samples L are very similar to each other. The samples belonging to class V are not that similar among themselves. That is why they are assigned to the darkest neurons in the map.

The map illustrates how some elements of different classes can be similar to each other. In effect, even though most of the neurons labeled as the same class are concentrated in the same zones, some neurons "stand" in the middle of neurons belonging to other classes. For example, in the middle of the group of neurons belonging to class V on the left bottom of the map, there is a neuron labeled L. The samples mapped to this neuron are not very similar among them but they are closer to samples class V than to class L.

Figure 4-25 complements this information with the entropy of each neuron.

Most of the neurons have zero entropy. Those few neurons with high entropy are located in the boundaries of the clusters. They are shown dark in the cluster map. The high entropy is due to the fact that some samples of different classes can be similar to each other, particularly in the boundaries of the clusters. For example, in Figure 4-25, there is a cluster class V on the top left, a cluster R on its right, and some sparse neurons assigned to class N. Some neurons in between these clusters have high entropy:

- Class L with entropy of 0.97 (2V and 3L)
- Class V with entropy 0.81 (3V and 1N)
- Class V with entropy of 1 (1V and 1L)
- Class V with entropy of 1.58 (1V, 1L and 1N)

Some neurons (most of the darkest ones) do not have any samples mapped to them. These are mainly in the boundary of the brighter zones and could be used to separate the clusters. There is a drawback about these neurons. If any of the samples of the testing set is mapped to a neuron that was empty in training, its class will be unknown.

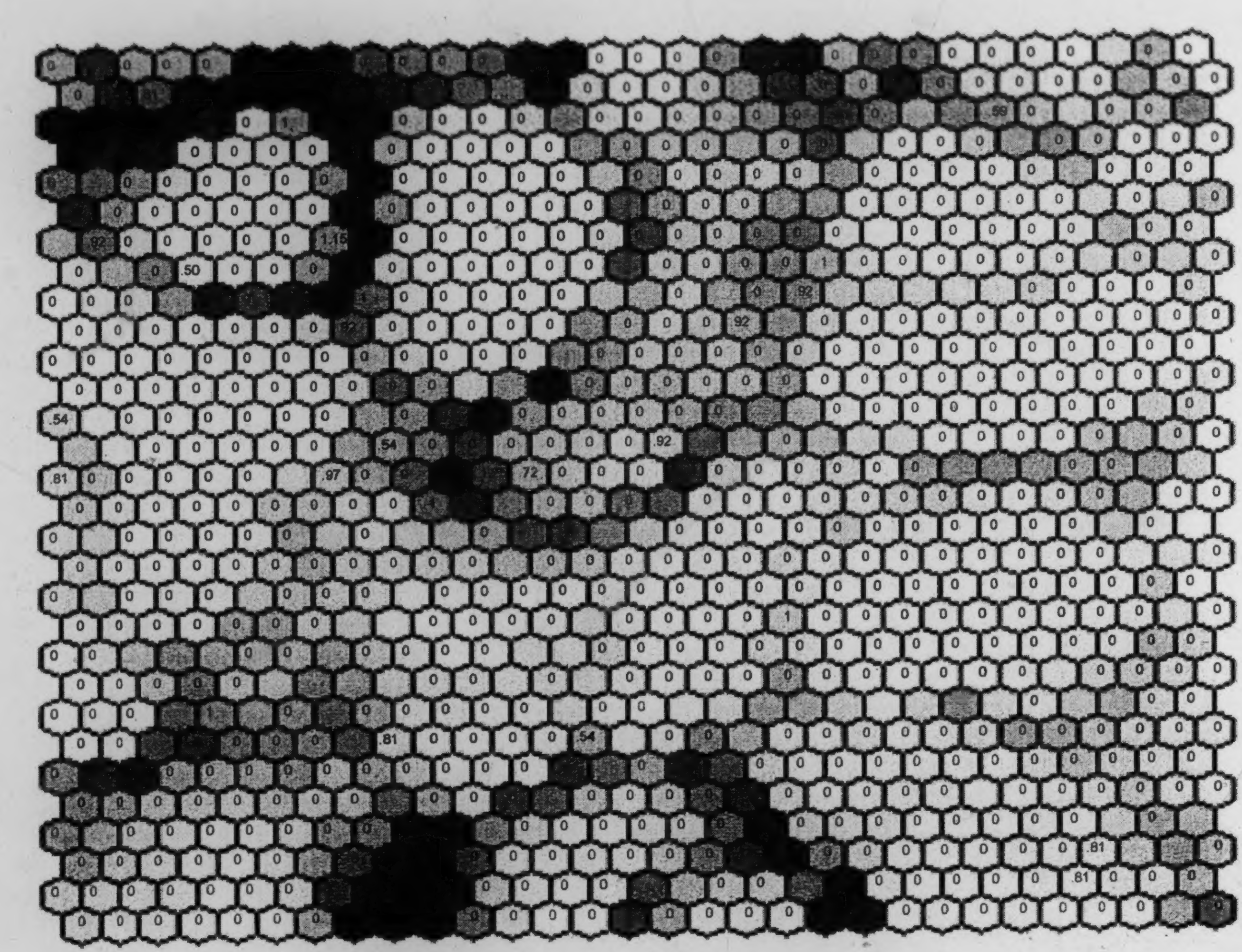


Figure 4-25: SOM classifier 30x30. Cluster map with entropy per neuron
(Class N in blue, class V in red, class L in black and class R in green).

The confusion matrices for training and testing of the SOM classifier with the winning configuration are shown below:

Table 4-19: Confusion matrix for training and testing using the SOM classifier 30x30 (QRS segments with logistic normalization)

Training				
Out\Real	V	R	L	N
V	**486**	2	6	8
R	3	**852**	0	4
L	4	0	**946**	4
N	8	1	3	**857**
Accuracy = 0.9865				
Kappa score = 0.9817				

Testing				
Out\Real	V	R	L	N
V	**309**	1	3	5
R	6	**551**	0	1
L	4	0	**615**	2
N	5	2	4	**548**
Accuracy = 0.9516				
Kappa score =: 0.9358				

Observations on the confusion matrices (Table 4-19):

The accuracy and kappa score in training are quite acceptable. In testing, the accuracy and kappa score are slightly smaller. Most of the misclassified patterns in this table belong to class V, the class with fewer patterns in the set. On the other hand, there was less error on class L patterns, the class with the most patterns in the set.

In neither training nor testing, were the patterns L and R mistaken with one another. The main confusions in the classifications were between classes V and N and between classes V and L. The reason for this could be that depending on where the premature ventricular block occurred, the QRS class V could bear a resemblance to a class R (if originated from the left ventricle), a class L (if originated from the right ventricle) or even to a class N (if originated from the bundle of His).

4.4.2. RBF-OLS

A. Architecture and Methodology

Radial Basis Functions neural networks (RBF-NN) are used to classify the data set formed by QRS segments described in Section 2.6.3 and also on its transformed version using 10 Hermite coefficients.

The data set was divided in the same way as for the experiments with SOM: 60% of each class in the training set and 40% in the testing set.

The orthogonal least squares method (OLS) is used to determine the centers of the network as explained in Section 3.4.

The configuration of the RBF-OLS network is the following:

- Number of centers. Set arbitrarily to:
  - 4
  - 250, an intermediate value
  - 900, corresponding to the maximum number of neurons in the experiments with SOM.
- Type of basis function: Gaussian.
- Spread: Mean Standard deviation for all the variables of the training data.
- Normalization: None, logistic.

B. Results and observations

The results are shown in the form of tables showing the confusion matrices, accuracy and kappa score for training and testing for 4, 250 and 900 centers. There are 4 tables: one for each type of data and normalization.

Figure 4-26 and Figure 4-27 summarize the results for training and for testing.

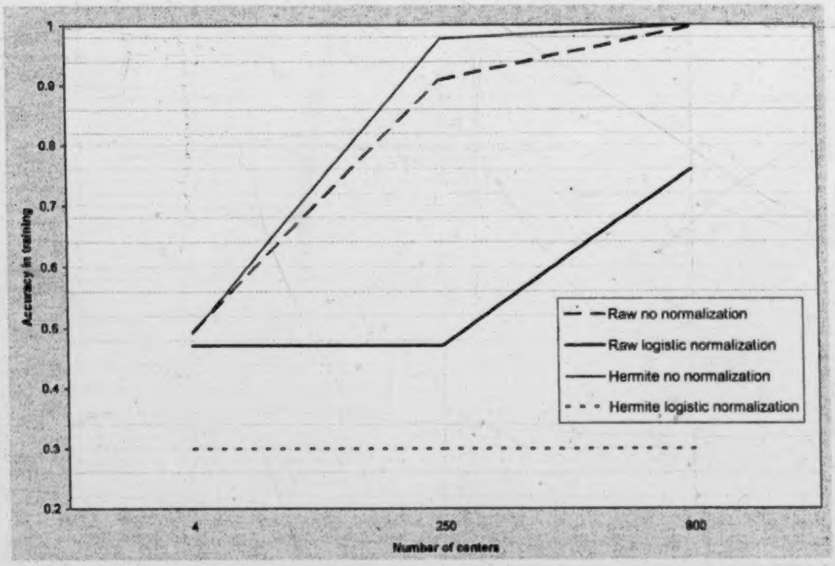


Figure 4-26: Accuracy in training of RBF-OLS classifier depending on the number of centers.

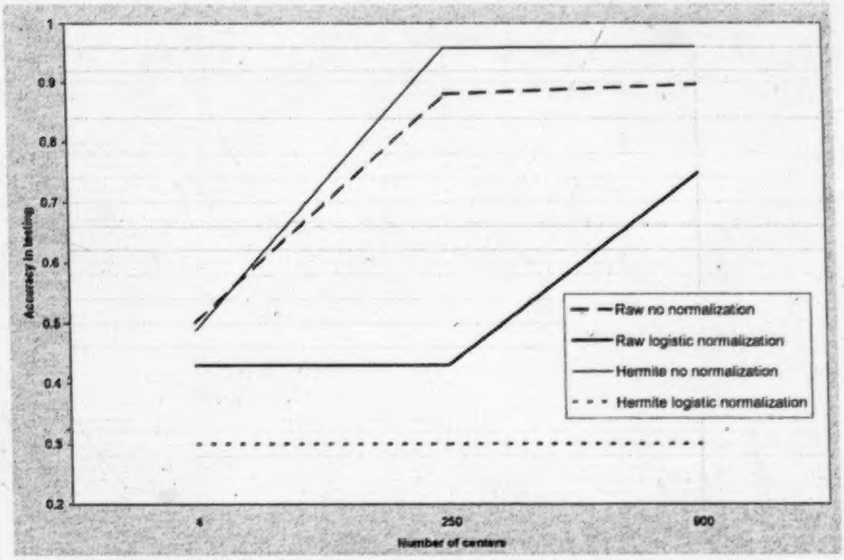


Figure 4-27: Accuracy in testing of RBF-OLS classifier depending on the number of centers

Irrespective of the number of centers of the RBF, the best results are obtained in both training and testing when the classification is made on the Hermite coefficients with no normalization. The best accuracy is reached with 900 centers: 1 in training and 0.9614 in testing.

The logistic normalization does not improve the performance of the classifier in any of the cases.

As expected, the number of centers affects the accuracy of the classifier in training: the more centers, the higher the accuracy. This is not true for the testing, in which case the accuracy does not improve significantly for more than 250 centers. This might mean that the number of centers affects the generalization capability of the classifier.

Table 4-20: Results using the RBF-OLS classifier with 4, 250 and 900 centers on the QRS segments with no normalization

4 Centers									
Training					**Testing**				
Out\Real	V	R	L	N	Out\Real	V	R	L	N
V	**0**	0	0	0	V	**0**	0	0	0
R	0	**189**	0	0	R	0	**130**	0	0
L	1	0	**518**	0	L	1	0	**352**	0
N	500	666	437	**873**	N	333	441	286	**583**
Accuracy in training = 0.4962					Accuracy in testing = 0.5009				
Kappa score in training = 0.3022					Kappa score in testing = 0.3086				

250 Centers									
Training					**Testing**				
Out\Real	V	R	L	N	Out\Real	V	R	L	N
V	**496**	128	71	83	V	**329**	107	48	89
R	0	**726**	0	1	R	1	**464**	0	3
L	1	0	**884**	1	L	2	0	**589**	0
N	4	1	0	**788**	N	2	0	1	**491**
Accuracy in training = 0.9089					Accuracy in testing = 0.8810				
Kappa score in training = 0.8783					Kappa score in testing = 0.8416				

900 Centers									
Training					**Testing**				
Out\Real	V	R	L	N	Out\Real	V	R	L	N
V	**492**	0	0	0	V	**168**	0	1	0
R	0	**855**	0	0	R	2	**555**	0	3
L	0	0	**955**	0	L	2	0	**605**	1
N	9	0	0	**873**	N	162	16	32	**579**
Accuracy in training = 0.9972					Accuracy in testing = 0.8970				
Kappa score in training = 0.9962					Kappa score in testing = 0.8587				

Observations on Table 4-20: Results using the RBF-OLS classifier with 4, 250 and 900 centers on the QRS segments with no normalization.

If only four centers are used in the RBF-OLS to classify the QRS segments with no normalization, practically all the segments that are not assigned to the correct class are assigned to class N, especially the ones that belong to class V. This happens because of the OLS strategy to choose the centers that first selects the samples for which the squared error is smaller. The squared error is usually smaller for the samples belonging to the dominant classes in the data set. The value of the kappa score emphasizes this fact, being so much smaller than the value of the accuracy.

With 250 centers, the accuracy is more acceptable. It is interesting to note that when the output of the classifier was classes R, L or N, most of the cases were correct. But this was not the case for class V. Furthermore, about a quarter of the data was assigned to each class. If more insight were needed, it would be probably interesting to see to which classes the centers belong.

When the RBF-OLS classifier is built with 900 centers, the training data is almost perfectly classified but the testing results are poor, loosing, as in the case of the SOM 30x30 its capability of generalization.

Observations on Table 4-21: Results using the RBF-OLS classifier with 4, 250 and 900 centers on the QRS segments with logistic normalization.

Logistic normalization deteriorates the results obtained with the RBF-OLS classifier on the QRS segments. In this case, with 250 centers the classifier does not raise its accuracy above 50%, and even with 900 centers it can't increase its accuracy above 80%. Most of the data are assigned to class L with this configuration.

Table 4-21: Results using the RBF-OLS classifier with 4, 250 and 900 centers on the QRS segments with logistic normalization

4 Centers									
Training					Testing				
Out\Real	V	R	L	N	Out\Real	V	R	L	N
V	**0**	0	0	0	V	**141**	0	0	0
R	0	**242**	0	0	R	0	**0**	0	0
L	501	613	**955**	568	L	0	0	**137**	0
N	0	0	0	**305**	N	442	334	434	**638**
Accuracy in training = 0.4717					Accuracy in testing = 0.4309				
Kappa score in training = 0.2506					Kappa score in testing = 0.1912				

250 Centers									
Training					Testing				
Out\Real	V	R	L	N	Out\Real	V	R	L	N
V	**0**	0	0	0	V	**0**	0	0	0
R	0	**242**	0	0	R	0	**137**	0	0
L	501	613	**955**	568	L	334	434	**638**	442
N	0	0	0	**305**	N	0	0	0	**141**
Accuracy in training = 0.4717					Accuracy in testing = 0.4309				
Kappa score in training = 0.2506					Kappa score in testing = 0.1912				

900 Centers									
Training					Testing				
Out\Real	V	R	L	N	Out\Real	V	R	L	N
V	**350**	0	7	9	V	**217**	2	5	6
R	28	**661**	9	59	R	31	**442**	7	43
L	85	101	**831**	229	L	60	60	**563**	167
N	38	93	108	**576**	N	26	67	63	**367**
Accuracy in training = 0.7594					Accuracy in testing = 0.7474				
Kappa score in training = 0.6707					Kappa score in testing = 0.6537				

Table 4-22: Results using the RBF-OLS classifier with 4, 250 and 900 centers on the Hermite coefficients data with no normalization

4 Centers										
Training						**Testing**				
Out\Real	V	R	L	N		Out\Real	V	R	L	N
V	**0**	0	0	0		V	**0**	0	0	0
R	482	**771**	444	491		R	326	**529**	310	335
L	14	35	**470**	60		L	8	11	**302**	41
N	5	49	41	**322**		N	0	31	26	**207**
Accuracy in training = 0.4909						Accuracy in testing = 0.4882				
Kappa score in training = 0.2978						Kappa score in testing = 0.2945				

250 Centers										
Training						**Testing**				
Out\Real	V	R	L	N		Out\Real	V	R	L	N
V	**487**	14	19	20		V	**317**	14	14	25
R	2	**840**	0	1		R	5	**555**	0	5
L	3	0	**936**	3		L	7	0	**619**	4
N	9	1	0	**849**		N	5	2	5	**549**
Accuracy in training = 0.9774						Accuracy in testing = 0.9595				
Kappa score in training = 0.9694						Kappa score in testing = 0.9453				

900 Centers										
Training						**Testing**				
Out\Real	V	R	L	N		Out\Real	V	R	L	N
V	**501**	0	0	0		V	**278**	0	3	7
R	0	**855**	0	0		R	45	**568**	1	7
L	0	0	**955**	0		L	7	0	**632**	3
N	0	0	0	**873**		N	4	3	2	**566**
Accuracy in training = 1						Accuracy in testing = 0.9614				
Kappa score in training = 1						Kappa score in testing = 0.9476				

Table 4-23: Results using the RBF-OLS classifier with 4, 250 and 900 centers on the Hermite Coefficients data with logistic normalization

4 Centers										
Training						**Testing**				
Out\Real	V	R	L	N		Out\Real	V	R	L	N
V	**1**	0	0	0		V	**1**	0	0	0
R	0	**0**	0	0		R	0	**0**	0	0
L	500	855	**955**	873		L	333	571	**638**	583
N	0	0	0	**0**		N	0	0	0	**0**
Accuracy in training = 0.3003						Accuracy in testing = 0.3006				
Kappa score in training = 0.0005						Kappa score in testing = 0.0008				

250 Centers										
Training						**Testing**				
Out\Real	V	R	L	N		Out\Real	V	R	L	N
V	**1**	0	0	0		V	**1**	0	0	0
R	0	**0**	0	0		R	0	**0**	0	0
L	500	855	**955**	873		L	333	571	**638**	583
N	0	0	0	**0**		N	0	0	0	**0**
Accuracy in training = 0.3003						Accuracy in testing = 0.3006				
Kappa score in training = 0.0005						Kappa score in testing = 0.0008				

900 Centers										
Training						**Testing**				
Out\Real	V	R	L	N		Out\Real	V	R	L	N
V	**1**	0	0	0		V	**1**	0	0	0
R	0	**0**	0	0		R	0	**0**	0	0
L	500	855	**955**	873		L	333	571	**638**	583
N	0	0	0	**0**		N	0	0	0	**0**
Accuracy in training = 0.3003						Accuracy in testing = 0.3006				
Kappa score in training = 0.0005						Kappa score in testing = 0.0008				

Observations on Table 4-22: Results using the RBF-OLS classifier with 4, 250 and 900 centers on the Hermite coefficients data with no normalization.

The transformation to Hermite coefficients clearly improves the performance of the RBF-OLS classifier.

As was expected, with four centers the accuracy was low. It does not recognize any of the samples belonging to class V and most of the samples are classified as R.

When 250 centers are used the accuracy and the kappa score improve considerably. Using 900 centers improves the training accuracy to a perfect result of 1, but the testing results are not better than with 250 centers. The capability to recognize samples belonging to class V deteriorates when 900 centers are used in the RBF-OLS classifier.

Observations on Table 4-23: Results using the RBF-OLS classifier with 4, 250 and 900 centers on the Hermite Coefficients data with logistic normalization.

The data of Hermite coefficients with logistic normalization is clearly not separable using RBF-OLS classifier, whatever number of centers was used to do so.

4.4.3. LDA

A. Architecture and Methodology

LDA is a classification method explained in Section 3.2.

The training process was performed on 60% of the data set, while the testing was performed on the remaining 40%. Three data sets were used: the data set formed by QRS segments described in Section 2.6.3, its transformed version using 10 Hermite coefficients, and the data set generated with the PCA (using 15, 20, 50 principal components).

No normalization of the data was performed other than the normalization inherent to the LDA.

B. Results and observations

Table 4-24: Accuracy with the LDA classifier

Data Set	Accuracy Training	Accuracy testing
QRS segments	0.9111	0.9124
Hermite coefficients	0.7249	0.7079
PCA, 15 principal components	0.8650	0.8542

With the LDA, the untransformed QRS segments are better classified than any of the implemented transformations.

4.4.4. Proposed classifier SOM-RBF

A. Architecture and Methodology

The training process was performed on 60% of the data set, while the testing was performed on the remaining 40%. Two data sets were used: the data set formed by QRS segments described in Section 2.6.3 and its transformed version using 10 Hermite coefficients. The normalization of the data set was either none or logistic.

With the proposed SOM-RBF classifier, the centers of the RBF classifier were obtained training the Self Organizing Maps, determining the clusters into which the data could be divided, and calculating the centers of these clusters.

Similarly to Section 4.4, the SOM was trained and displayed using the tool presented by Bai in [5] with the following configuration:

- the measure of distance is Euclidean.
- the neighborhood kernel is Gaussian.
- the size of the neighborhood is initially 3, with a decline rate of 0.1.
- the neurons have hexagonal shape.
- the number of epoch for the rough training is 5000.
- the number of epoch for the fine-tuning of the training is 10000.
- the learning rate is variable:
  - during the rough training, it starts at 1 and decreases to 0.2 throughout the mentioned number of epoch.
  - during the fine-tuning, it starts at 0.2 and ends at 0.01.
- the Initialization of the neurons is random.
- the different sizes of the maps that were tried are shown in Table 4-25.

When determining the final clusters using the results from the SOM and the knowledge of the data (i.e. the class to which each sample belong), the following criteria were used:

- Initial gray level threshold, above which a neuron is assigned to a cluster: The values are shown in Table 4-25.
- Maximum acceptable entropy for a cluster: 0.1.
  (If the entropy of a cluster formed by several neurons is higher, the cluster is split).
- Minimum amount of samples in a cluster to be taken into account: 10.

The parameters of the RBF neural network are the following:

- Type of basis function: Gaussian.
- Normalization none, or logistic. Same normalization as in the clustering process with the SOM.
- Number of centers: Determined by the previous stage.
- Initial spread: Given by the previous stage (Refer to Table 4-25).
- Learning rate for the gradient descent search for an optimal spread: 0.1

Several heuristics were tried to optimize the performance of the classifier. Each parameter was controlled by one variable in the software. They are presented below in Table 4-25. The list of parameters is divided in three parts: training of the SOM, generating the clusters, and configuring the actual RBF classifier.

Table 4-25: Parameters adjusted on the proposed SOM-RBF classifier

Stage of the process	Parameter	Value of variable	Meaning
Training the SOM	Size of the SOM (Variable ms_index)	1	05 neurons by 05 neurons
		2	10 neurons by 10 neurons
		3	15 neurons by 15 neurons
		4	20 neurons by 20 neurons
		5	25 neurons by 25 neurons
		6	30 neurons by 30 neurons
Generating the final clusters from the SOM and with the information of classes.	Include or not the darker neurons as individual clusters (Variable yesnodark)	1	yes
		2	no
	Minimum amount of data a neuron needs to be included in the cluster determination (Variable dt_index)	1	1% maximum amount of data in neurons
		2	2% maximum amount of data in neurons
		3	5% maximum amount of data in neurons
	Initial gray level threshold on the cluster map to separate the clusters. (Variable t_index)	1	0 (The whole map is initially one cluster)
		2	50
		3	100
		4	255 (Each neuron is a cluster)
Finding the centers	From which values the center is calculated (Variable whosecenternum)	1	From values of data in cluster
		2	From weights of neurons in the cluster
	Type of center (Variable centertypenum)	1	Mean of each variable of the data in the cluster
		2	Median of each variable of the data in the cluster
Configuring the RBF	How the initial spread is estimated (Variable opspreadnum)	1	Five multiples of the average standard deviation of the data in the cluster
		2	Average Euclidean distance from the data in the cluster to its center + Five multiples of its standard deviation
	Initial spread for the gradient descent search of optimal spread for the RBF classifier. Taken from the values in opspread. (Variable opspreadnum)	1	First multiple
		2	Second multiple
		3	Third multiple
		4	Fourth multiple
		5	Fifth multiple

B. Results

The best result for the SOM-RBF classifier is given in Table 4-26. It was obtained on the data set formed by QRS segments with no normalization using the following configuration:

- The centers used were the median of the weights of the neurons of the SOM for each cluster
- The spread was three times the average per feature of the standard deviation of the data in each cluster.
- Dark neurons were included as clusters
- The minimum amount of data to include a neuron was irrelevant (dt_index=1,2,3 gave the same result).
- Each acceptable neuron was a cluster by itself.

Table 4-26: Best result for the SOM-RBF classifier

Number of centers = 114										
Training						**Testing**				
Out\Real	V	R	L	N		Out\Real	V	R	L	N
V	**464**	3	1	2		V	**298**	0	1	3
R	6	**841**	0	2		R	6	**561**	0	0
L	14	1	**947**	16		L	21	0	**631**	14
N	17	10	7	**853**		N	9	10	6	**566**
Accuracy in training = 0.9752						Accuracy in testing = 0.9671				
Kappa score in training = 0.9663						Kappa score in testing = 0.9553				

The best results by type of data, type of normalization, and by size of the SOM are presented in Figure 4-28 and Figure 4-29:

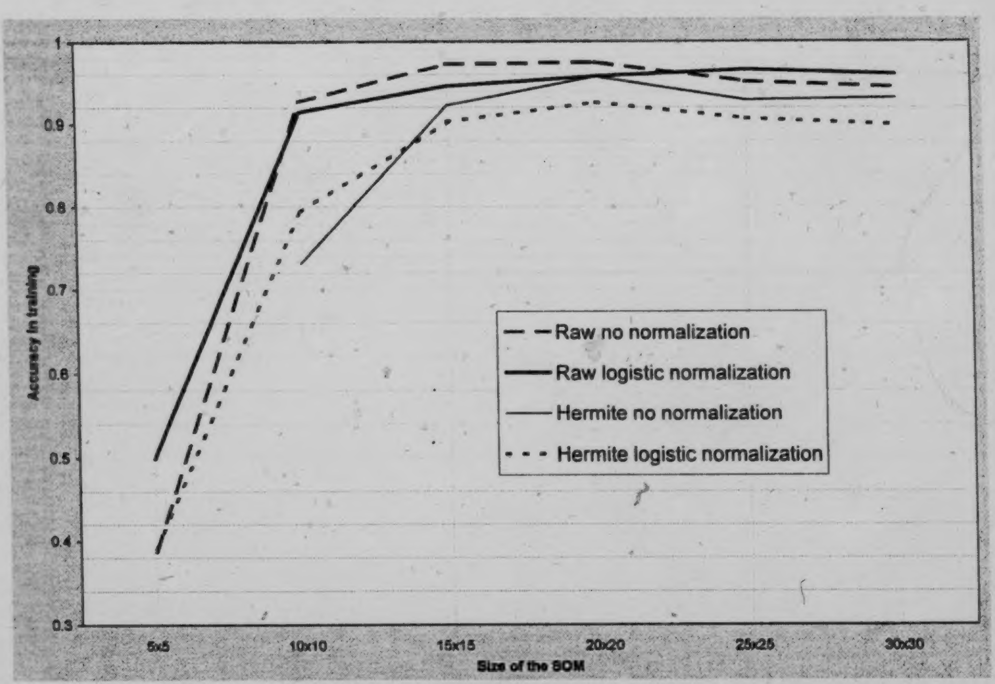


Figure 4-28: Best results in training of SOM-RBF classifier depending on the size of the SOM.

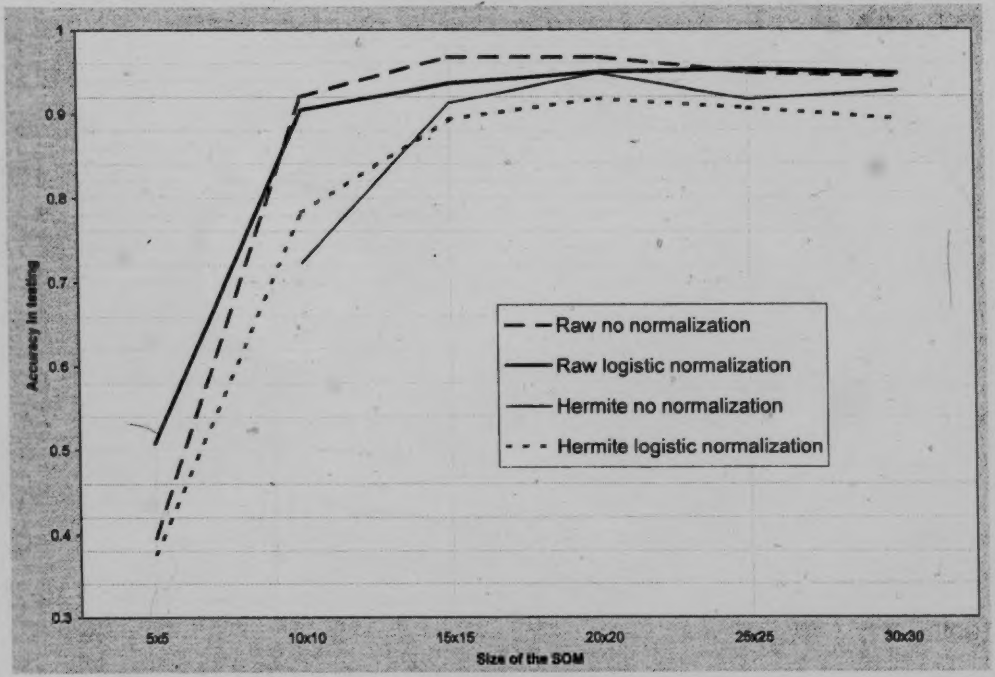


Figure 4-29: Best results in testing of SOM-RBF classifier depending on the size of the SOM.

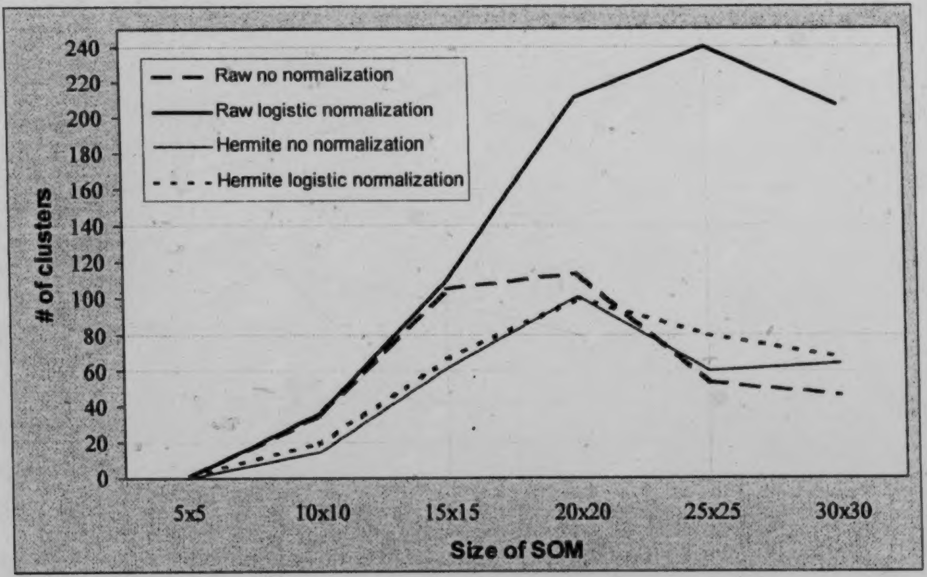


Figure 4-30: Number of clusters vs. size of the SOM

The values that generated the previous chart are presented below in Table 4-27, Table 4-28, Table 4-29, and Table 4-30.

Table 4-27: Best results of SOM-RBF for the QRS segments with no normalization.

Size of SOM	5x5	10x10	15x15	20x20	25x25	30x30
Accuracy in training	0.3869	0.9268	0.9730	0.9752	0.9516	0.9447
Accuracy in testing	0.3960	0.9205	0.9675	0.9671	0.9483	0.9440
Kappa score in training	0.1551	0.9001	0.9633	0.9663	0.9342	0.9248
Kappa score in testing	0.1680	0.8914	0.9559	0.9553	0.9296	0.9238

Table 4-28: Best results of SOM-RBF for the QRS segments with logistic normalization

Size of SOM	5x5	10x10	15x15	20x20	25x25	30x30
Accuracy in training	0.4994	0.9143	0.9460	0.9579	0.9664	0.9604
Accuracy in testing	0.5085	0.9045	0.9356	0.9492	0.9520	0.9473
Kappa score in training	0.3207	0.8845	0.9269	0.9429	0.9545	0.9463
Kappa score in testing	0.3334	0.8713	0.9128	0.9310	0.9350	0.9285

Table 4-29: Best results of SOM-RBF for the Hermite coefficients data set with no normalization

Size of SOM	5x5	10x10	15x15	20x20	25x25	30x30
Accuracy in training	Not Available	0.7318	0.9224	0.9576	0.9290	0.9318
Accuracy in testing		0.7230	0.9125	0.9478	0.9167	0.9271
Kappa score in training		0.6355	0.8945	0.9424	0.9036	0.9073
Kappa score in testing		0.6237	0.8811	0.9290	0.8868	0.9008

Table 4-30: Best results of SOM-RBF for the Hermite coefficients data set with logistic normalization

Size of SOM	5x5	10x10	15x15	20x20	25x25	30x30
Accuracy in training	0.3910	0.7940	0.9026	0.9259	0.9067	0.8992
Accuracy in testing	0.3763	0.7827	0.8932	0.9172	0.9059	0.8932
Kappa score in training	0.1683	0.7203	0.8686	0.8997	0.8740	0.8639
Kappa score in testing	0.1449	0.7049	0.8558	0.8879	0.8730	0.8558

Several combinations of the parameters shown in Table 4-25 gave the same results in the classification process. These combinations are summarized in Table 4-31. It can be observed in this table that the following values of variables generate the best results in most of the cases:

- Variable whosecenter=2. The centers are obtained from the weights of the neurons in the SOM.
- Variable opspread=1. The spread is a multiple of the standard deviation of the data in the cluster.
- Variable t_index=4. Each acceptable neuron is a cluster.

Table 4-31: Value of the parameters that produced the best classification results.

Data Set	Normalization	Size SOM	#centers	whosecenter	centertype	opspread	yesnodark	yesnosplit	dt_index	t_index	I_spreadops
Raw	None	5x5	1	1	1	1	1	1	1.2.3	1.2.3.4	3
		10x10	34	2	2	1	1	1	1.2.3	4	3.4
		15x15	105	2	1.2	2	1	1	1.2.3	4	1.2.3.4.5
		20x20	**114**	**2**	**2**	**1**	**1**	**1**	**1.2.3**	**4**	**3**
		25x25	53	2	2	1	1	1	1.2.3	4	3
		30x30	46	2	2	1	1	1	3	2.3	5
	Logistic	5x5	1	2	1	1	1	1	1.2.3	1.2.3.4	4
		10x10	36	2	1.2	2	1	1	1.2	4	1.2.3.4.5
		15x15	110	2	2	1	1	1	1.2.3	4	2
		20x20	211	2	2	1	1	1	1.2.3	4	4
		25x25	240	2	1	1	1	1	1-3	4	5
		30x30	207	2	2	1	1	1	1-2	4	3
Hermite Coefficients	None	5x5	Not Available (the entropy of all the neurons was below the threshold of acceptable entropy)								
		10x10	14	2	2	1	1	1	1.2.3	4	5
		15x15	61	2	2	1	1	1	1.2.3	4	2
		20x20	101	2	2	1	1	1	1.2.3	4	2
		25x25	60	2	2	2	1	1	3	1.2.3	1.2.3.4.5
		30x30	64	2	2	1	1	1	1-2	1.2	3
	Logistic	5x5	1	2	2	1	1	1	1.2.3	1.2.3.4	3
		10x10	19	2	1	1	1	1	1.2.3	4	3
		15x15	66	2	2	1	1	1	1.2.3	4	1
		20x20	100	2	2	1	1	1	1.2.3	4	2
		25x25	80	2	2	1	1	1	1.2.3	4	5
		30x30	67	2	2	1	1	1	3	1	2

Figure 4-31 and Figure 4-32 below show the clusters used to generate the centers of the RBF. The number next to the label is the identification given to the cluster by the software. Many of the neurons do not belong to any clusters even though visually they ought to be. This is due to the number of samples mapped to it (refer to Figure 4-32). In effect, the software was arbitrarily configured to accept only neurons with more than ten samples mapped to it. In Figure 4-32, some neurons are highlighted that could belong to new clusters if the minimum number of samples accepted per neuron was decreased. These neurons might be important for improving the results because their class label is different from those of the neurons that surround them.

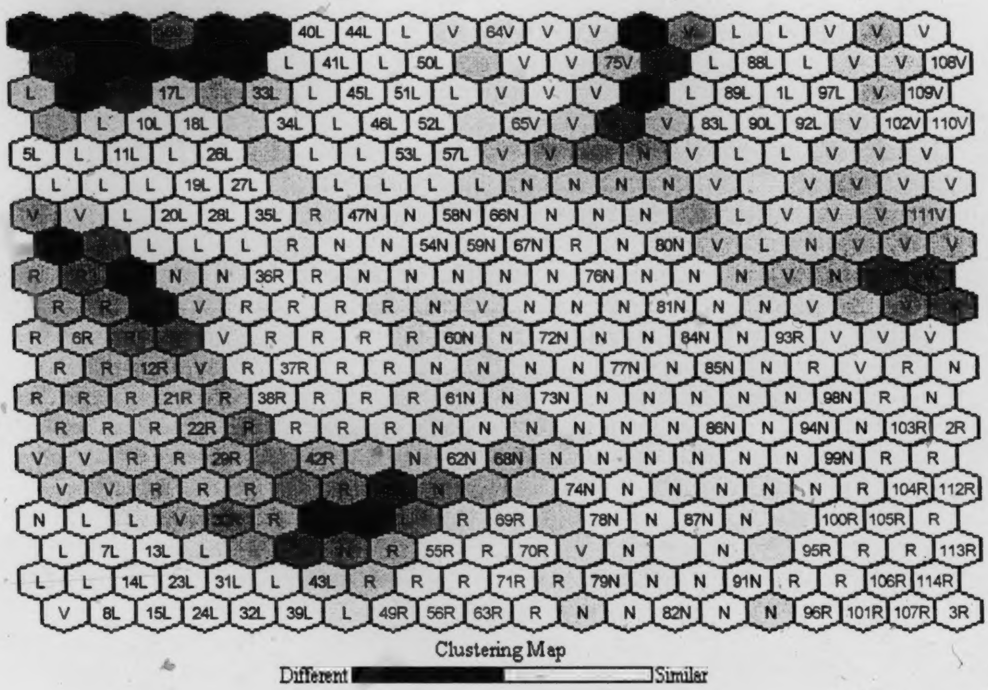


Figure 4-31: clustering map of the SOM 20x20, with the winning class per neuron, and the cluster number assigned by the software.

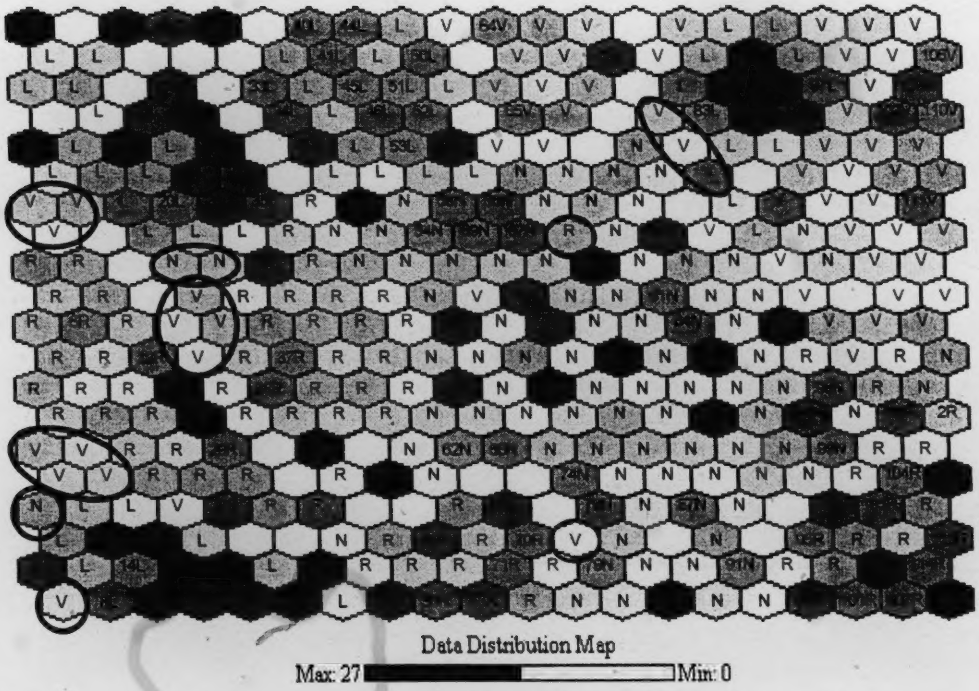


Figure 4-32: Distribution map of the SOM 20x20, with the winning class per neuron, and the cluster number assigned by the software

4.4.5. Summary of the best classification results

The following table shows the best results obtained with each method of classification implemented in this work.

Table 4-32: Summary of the best classification results

	Classification method			
	LDA	SOM 900 neurons	RBF-OLS 900 centers	SOM-RBF 114 centers
Data Set	Raw	Raw	Hermite coefficients	Raw
Normalization	None	Logistic	None	None
Accuracy in training	0.9111	0.9865	1	0.9752
Accuracy in testing	0.9124	0.9516	0.9614	0.9671
Kappa score in training		0.9817	1	0.9663
Kappa score in testing		0.9358	0.9476	0.9553

The best accuracy is obtained with the SOM classifier. In particular, one can see that the proposed algorithm (SOM-RBF) gives the best result in terms of accuracy and kappa score in testing. The RBF-OLS gave excellent results in training but its performance in testing did not surpass the results of the SOM-RBF.

5. Conclusion and Recommendations for future work

Several techniques were exploited for QRS detection, feature extraction, and classification of cardiac beats from digital electrocardiograms obtained from one lead from the MIT-BIH database [53].

Digital filtering, non-linear transform, and neural networks configuration were implemented and the results showed that digital filtering can be an excellent tool for detecting QRS, given that the thresholding process is adequate. The advantage of the neural network over the other algorithms (in addition to achieving good results) is its ability to learn from the data. This advantage could be introduced in future work to the Okada algorithm, for example, by introducing a neural network whose inputs are the pulses to be passed though the thresholding process. Genetic algorithms could be used to optimize the values of the adjustable parameters of the algorithm.

Feature extraction consisted of Hermite transformation and principal component analysis. In the first case, the number of features to use in the classification process was selected based on: 1)the approximation capability of their inverse, and 2)the accuracy with an LDA classifier. In the second case (the PCA), another factor was also taken into account: the variance contribution of the features to the total variance of the whole data set.

The classification results were satisfactory, mostly for the experiments performed with the QRS segments. The positive aspect of this situation is that no complicated processing was carried out in order to produce good results. The drawback to this situation is that the data set of QRS segments is constructed into a highly dimensional feature space. A recommendation for future work would be to design a re-sampling algorithm that would reduce the number of variables to process while keeping the information needed to discriminate among classes.

It was observed that this field of ECG signal analysis and classification is open to many innovations: from introducing a learning strategy in determining the heuristics of the QRS detectors to finding a set of features with the appropriate classifier to enable the correct discrimination of patterns.

In this work, QRS classification was performed independently of QRS detection. A recommendation for future investigation would be to integrate both steps by designing classifiers that are robust to changes in the position of the R wave in the segments that are to be classified.

In this study, just as in the majority of the reviewed literature, only one channel from the electrocardiogram signal was used. However, the cardiologists use several channels to assess their diagnosis. Therefore, the automatic analysis of the ECG should be done using more channels, taking into account that not only more information would be introduced but also more noise.

6. Bibliography

[1] V. X. Afonso, "ECG QRS Detection," *Biomedical Digital Signal Processing: C-Language Examples and Laboratory Experiments for the IBM PC,* W. J. Tompkins, ed., pp. 236-280, Upper Saddle River, NJ: Prentice Hall Inc., 1995.

[2] V. X. Afonso, W. J. Tompkins, T. Q. Nguyen, and S. Luo, "ECG Beat Detection Using Filter Banks," *IEEE Transactions on Biomedical Engineering,* vol. 46, no. 2, pp. 192-202.

[3] V. A.. Allen, J. Belina, "ECG Data Compression using the Discrete Cosine Transform (DCT)," *Proceedings in Computers in Cardiology*, pp.687-690. Durham, North Carolina: IEEE Computer Society Press Los Alamitos, California, 1992.

[4] T. Bäck, *Evolutionary Algorithms in Theory and Practice.* New York: Oxford University Press, 1996.

[5] X. Bai. *Analysis of Software Measures by Self Organizing Maps. MSc. Thesis.* Alberta, Canada: University of Alberta, 2001.

[6] C. M. Bishop, *Neural Networks for Pattern Recognition.* Oxford: Clarendon Press, 1995.

[7] T. M. Blake, *Annotated Atlas of Electrocardiography. A guide to confident interpretation.* New Jersey: Humana Press Inc., 1999.

[8] G. Carrault, M. O. Cordier, R. Quiniou, F. Wang, "Temporal Abstraction and Inductive Logic Programming for Arrhythmmia Recognition from Electrocardiograms," *Artificial Intelligence in Medicine*, vol. 28, pp. 231-263, 2003.

[9] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 302-309, 1991.

[10] K. Cios, W. Pedrycz and R. Swiniarski, *Data Mining. Methods for Knowledge Discovery.* Boston: Kluwer Academic Publishers, 1998.

[11] D. Cuesta-Frau, J. C. Pérez-Cortés, and G. Andreu-García, "Clustering Of Electrocardigraph Signals In Computer-Aided Holter Analysis," *Computer Methods and Programs in Biomedicine*, vol. 100, pp. 1-18, 2003.

[12] W. W. Dai, Z. Yang, S. L Lim, O. Mikhailova, and J. Chee, "Processing and Analysis of ECG Signal Using Non-orthogonal Wavelet Transform," in *Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 20, no. 1, pp. 139-142, 1998.

[13] D. Daultrey, *Concepts and Techniques in Modern Geography n.8: Principal component Analysis.* 1976.

[14] H. Dickhaus and H. Heinrich, "Classifying Biosignals with Wavelet Networks. A Method for Noninvasive Diagnosis," *IEEE Engineering in Medicine and Biology*, pp.103-111, September/October, 1996

[15] Z. Dokur, and T. Ölmez, "ECG Beat Classification by a novel hybrid neural network," *Computer Methods and Programs in Biomedicine*, vol. 66, pp. 167-181, 2001.

[16] R. C. Dubes, "Cluster Analysis and related issues," *Handbook of Pattern Recognition and Computer Vision.* C. H. Chen, and L. F. P. Wang eds., pp. 3-32, World Science Publishing Company: 1993.

[17] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* Toronto: John Wiley & Sons, Inc., 1973.

[18] R. O. Duda, P. E. Hart, D. G. Stork. *Pattern Classification.* Second ed., New York: John Wiley & Sons Inc., 2001.

[19] J. L. Fleiss, "Measuring Agreement Between Two Judges on the Presence or Absence of a Trait," *Biometrics*, vol. 31, pp. 651-659, 1975.

[20] G. E. Forsythe, M.A. Malcolm, C. B. Moler, *Computer methods for mathematical computations.* Englewood Cliffs, N.J.: Prentice-Hall, 1977.

[21] C. García-Berdonés, J. Narváez, U. Fernández, and F. Sandoval, " A New QRS Detector Based on Neural Network," in *Biological and Artificial Computation: From Neuroscience to Technology. International Work-Conference on Artificial and Natural Neural Networks, IWANN'97.* J. Mira, R. Moreno-Díaz and J. Cabestany, eds., Lanzalote, pp. 1260-1269, Canary Islands, Spain: June 4-6, 1997.

[22] H. Gholam-Hosseini, and H. Nazeran, "Detection and Extraction of the ECG Signal Parameters," in *Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 20, no. 1, pp. 127-130, 1998.

[23] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Don Mills, Ontario: Addison-Wesley, 1989.

[24] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C-K Peng, H. E. Stanley, "Physiobank, Physiotoolkit, and Physionet: Components of a New Research Resource for Complex Physiologic Signals," Circulation 101(23): e215-e220. [Circulation Electronic Pages; http://circ.ahajournals.org/cgi/content/full/101/23/e215]

[25] N. Goldschlager and M. J. Goldman, *Electrocardiography: Essentials of interpretation*. Los Altos, California: LANGE Medical Publications, 1984.

[26] G. Golub and W. Kahan, "Calculating the singular values and pseudoinverse of a matrix," *SIAM Numerical Analysis*, vol.2, no. 2, pp. 205-224, 1965.

[27] F. Griztadi, "Towards a generalized scheme for QRS detection in ECG waveforms," *Signal Processing*, vol. 15, pp. 183-192, 1988.

[28] F. Ham and S. Han, "Classification of Cardiac Arrhythmias using Fuzzy ARTMAP," *IEEE Transactions on Biomedical Engineering*, vol. 43, no. 4, pp. 425-430, April 1996.

[29] P.S. Hamilton and W. J. Tompkins, "Quantitative Investigation of QRS Detection Rules Using the MIT-BIH Arrhythmia Database," *IEEE Transactions on Biomedical Engineering*, vol. 33, no. 12, pp. 1157-1165, December 1986.

[30] E. Hernández and G. Weiss. *A First Course on Wavelets*. Boca Raton, Florida: CRC Press LLC, 1996.

[31] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Cambridge: MIT Press, 1975.

[32] Y. H. Hu, S. Palreddy, and W. J. Tompkins, "A patient-Adaptable ECG Beat Classifier Using a Mixture of Experts Approach," *IEEE Transactions on Biomedical Engineering*, vol. 44, no. 9, pp. 891-899, September 1997.

[33] R. J. Huszar, *Basic dysrhythmias. Interpretation and management*. St Louis: C.V. Mosby Company, 1988.

[34] A. Jain, *Algorithms for clustering data*. Englewood Cliffs, New Jersey: Prentice-Hall, 1988.

[35] J. D. Jobson. *Applied Multivariate Data Analysis. Volume II: Categorical and Multivariate Methods*. New York: Springer-Verlag New York Inc., 1992.

[36] S. Kadambe, R. Murray, and G. F. Boudreaux-Bartels, "Wavelet Transform-Based QRS complex detector," *IEEE Transactions on Biomedical Engineering*, vol. 46, no.7, pp.838-847, 1999.

[37] B-U. Köhler, C. Hennig, R. Orglmeister, "The Principles of Software QRS Detection: Reviewing and Comparing Algorithms for Detecting This Important Waveform," *IEEE Engineering in Medicine and Biology*, pp. 42-57, January/February 2002.

[38] T. Kohonen, *Self-Organization and Associative Memory*. second ed., New York: Springer-Verlag, 1988.

[39] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, pp. 59-69, 1982.

[40] M. Kundu, M. Nasipuri, and D. K. Basu, "A Knowledge-based Approach to ECG Interpretation Using Fuzzy Logic," *IEEE Transactions on Systems, Man, and Cybernetics part B*, vol. 28, no. 2, pp. 237-243, April 1998.

[41] M. Kundu, M. Nasipuri, and D. K. Basu, "Knowledge-Based ECG Interpretation: A Critical Review," *Pattern Recognition*, vol. 33, pp. 351-373, 2000.

[42] Kung S. Y. Kung, *Digital Neural Networks*. Englewood Cliffs New Jersey: Prentice Hall information and system sciences series, 1993.

[43] M. Lagerholm, C. Peterson, G. Braccini, L. Edenbrandt, and L. Sörnmo, "Clustering ECG Complexes Using Hermite Functions and Self-Organizing Maps," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 7, pp. 838-847, July 2000.

[44] P. Laguna, R. Jané, S. Olmos, N. V. Thakor, H. Rix, and P. Caminal, "Adaptive estimation of QRS complex wave features of ECG signal by the Hermite model," *Medical and Biological Engineering and Computing*, vol. 34, no. 1, pp. 58-68, January 1996.

[45] J. R. Landis, G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, pp. 159-174, 1977

[46] P.C. Li, N. Pizzi, W. Pedrycz, D. Westmore, and R. Vivanco, "Severe Storm Cell Classification Using Derived Products Optimized by genetic algorithms," in *Canadian Conference on Electrical and Computer Engineering*, Halifax, 2000.

[47] C. Li, C. Zheng and C. Tai, "Detection of ECG Characteristic Points Using Wavelet Transforms," *IEEE Transactions on Biomedical Engineering*, vol. 42, no.1, pp.21-28, January 1995

[48] N. Maglaveras, T. Stamkopoulos, K. Diamantaras, C. Pappas, and M. Strintzis, " ECG Pattern Recognition and Classification using Non-linear Transformations and Neural Networks: A review," *International Journal of Medical Informatics*, vol. 52, pp. 191-208, 1998.

[49] C. Maier, H. Dickhaus, and J. Gittinger, "Unsupervised Morphological Classification of QRS complexes," *Computers in Cardiology*, vol. 26, pp. 683-686, 1999.

[50] R.G. Mark, P.S. Schluter, G.B. Moody, P.H. Devlin, and D. Chernoff. An annotated ECG database for evaluating arrhythmia detectors. *Frontiers of Engineering in Health Care: Proceedings of the 4th Annual Conference of the IEEE Engineering in Medicine and Biology*, pp. 205-210. New York: IEEE Press, 1982.

[51] A. M. Martínez, A. C. Kak, "PCA versus LDA," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 228-233, February, 2001.

[52] J. Moody and C.J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation* vol. 1 no. 2, pp. 281-294, 1989.

[53] G.B. Moody, R.G. Mark. "The MIT-BIH Arrhythmia database on CD-ROM and software for use with it," *Computers in Cardiology*, vol. 17, pp. 185-188, 1990.

[54] M. E. Nygårds and L. Sörnmo, "Delineation of the QRS complex using the envelope of the ECG," *Medical & Biological Engineering & Computing*, vol. 21, pp. 538-547, 1983.

[55] M..Okada, "A Digital Filter for the QRS Complex Detection," *IEEE Transactions on Biomedical Engineering*, vol. 26, no.12, pp. 700-703, December 1979.

[56] M. I. Owis, A. H. Abou-Zied, M. Y. Abou-Bakr, and Y. M. Kadah, "Study of Features Based on Nonlinear Dynamical Modeling in ECG Arrhythmia Detection and Classification," *IEEE Transactions on Biomedical Engineering*, vol. 49, no. 7, pp. 733-736, July 2002.

[57] J. Pan and W. J. Tompkins, "A Real-Time QRS Detection Algorithm," *IEEE Transactions on Biomedical Engineering*, vol. 32, no. 3, pp. 230-236, March 1985.

[58] W. Pedrycz, "Conditional fuzzy clustering in the design of radial basis function neural networks," *IEEE Trans. Neural Networks*, vol. 9 no. 4, pp. 601-612, 1998.

[59] W. Pedrycz and F. Gomide, *An Introduction to Fuzzy Sets: Analysis and Design*. Cambridge: MIT Press, 1998.

[60] R. E. Phillips and M. K. Feeney, *The cardiac Rhythms. A systematic approach to interpretation*. Third ed., Philadelphia: W. B. Saunders Company, 1990.

[61] N.J. Pizzi, "Fuzzy Pre-processing of Gold Standards as Applied to Biomedical Spectra Classification," *Artificial Intelligence in Medicine*, vol. 16, pp. 171-182, 1999.

[62] M. J. D. Powell, "Radial Basis Functions for Multivariate Interpolation: A Review," *Algorithms for Approximations*. J. C. Mason and M. G. Cox, eds., pp. 143-167, Oxford: 1987.

[63] U. Rajendra Acharya, P. Subbanna Bhat, S. S. Iyengar, Ashok Rao, and Sumeet Dua, "Classification of heart rate data using artificial neural network and fuzzy equivalence relation," *Pattern Recognition*, vol. 36, pp. 61-68, 2003.

[64] C. R. Rao and S. K. Miltra, *Generalized Inverse of Matrices and Its applications*. New York: John Wiley, 1971.

[65] A. I. Rasiah, and Y. Attikiouzel, "A syntactic approach to the recognition of common cardiac arrhythmias within a single ambulatory ECG trace," *Australian Computer Journal*, vol. 26, no. 3, pp. 102-112, 1994.

[66] A. I. Rasiah, R. Togneri, and Y. Attikiouzel, "Modelling 1-D signals using Hermite basis functions," *IEE Proc.-Vis. Image Signal Process*, vol. 144, no. 6, pp.345-354, 1997.

[67] A.I. Rasiah, R. Togneri, and Y. Attikiouzel, "QRS Detection using Morphological and Rhythm information," *Proceedings of the IEEE* 0-7803-2768-3/95, pp. 2287-2293, 1995.

[68] D. H. Sanders, *Statistics. A Fresh Approach*. Fourth ed., Toronto: McGraw-Hill, Inc., 1990.

[69] R. Schalkoff, *Pattern Recognition. Statistical, Structural and Neural Approaches*. Toronto: John Wiley & Sons, Inc., 1992.

[70] L. Senhadji, G. Carrault, J. J. Bellanger, and G. Passariello, "Comparing Wavelet Transforms for Recognizing Cardiac Patterns," *IEEE Engineering and Biology*, vol. 14, no. 2, pp.167-173, March/April 1995.

[71] R. Silipo, W. Zong, M. Berthold, "ECG Feature Relevance in a Fuzzy Arrhythmia Classifier," *Computers in Cardiology*, vol. 26, pp. 679-682, 1999.

[72] L. Sörnmo, P.O. Börjesson, M. Nygårds, and O. Pahlm, "A method for evaluation of QRS Shape features using a Mathematical Model for the ECG," *IEEE transactions on Biomedical Engineering*, vol. 28, no. 10, pp. 713-717, October 1981.

[73] M. R. Spiegel, *Schaum's outline series. Theory and Problems of statistics*. New York: Schaum Publishing CO, 1961.

[74] StatSoft, Inc. (1999). *Electronic Statistics Textbook*. Tulsa, OK: StatSoft. WEB: http://www.statsoft.com/textbook/stathome.html.

[75] K. Sternickel, " Automatic Pattern Recognition in ECG Time Series," *Computer Methods and Programs in Biomedicine*, vol. 68, pp. 109-115, 2002.

[76] Y. Sun, S. Suppappola, T. A. Wrublewski, "Microcontroller-Based Real-Time QRS Detection," *Biomedical Instrumentation and Technology*, vol. 26, no.6, pp. 447-484, 1992.

[77] S. Suppappola and Y. Sun, "Nonlinear Transforms of ECG Signals for Digital QRS detection: A Quantitative Analysis," *IEEE Transactions on Biomedical Engineering*, vol. 41, no. 4, pp.397-400, April 1994.

[78] *The Oxford Dictionary of Current English*. Toronto: Oxford University Press, 1998.

[79] A. D. Timmis, *Cardiology*. Third ed., London: Mosby-Wolfe, 1995.

[80] J. Vesanto and E. Alhoniemi, "Clustering of the Self-Organizing Map," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp.586-600, May 2000.

[81] G. G. Walter, *Wavelets and Other Orthogonal Systems with Applications*. Boca Raton, Florida: CRC Press, Inc., 1994.

[82] H. Zeng, H. Wu, J. Lin, " QRS classification using adaptive Hermite decomposition and Radial Basis Function Network," in *Proceedings of the 20th annual conference of the IEEE Engineering in Medicine and Biology Society*, vol. 20, no. 1, pp. 147-150, 1998.

[83] W. Zong and D. Jiang, "Automated ECG Rhythm Analysis using Fuzzy Reasoning," *Computers in Cardiology*, vol. 25, pp. 69-72, 1998.